L Number	Hits	Search Text	DB	Time stamp
1	115	713/164.ccls.	USPAT	2003/11/12 17:38
2	150	713/165.ccls.	USPAT	2003/11/12 17:38
3	0	713.166.ccls.	USPAT	2003/11/12 17:38
4	83	713/151.ccls.	USPAT	2003/11/12 17:38
6	1	5164938.pn.	USPAT	2003/11/12 17:40
7	1	5935248.pn.	USPAT	2003/11/12 18:03
8	1	5689566.pn.	USPAT	2003/11/12 18:03
-	1090	713/201.ccls.	USPAT	2003/10/27 11:19
-	76	713/201.ccls. and "security level$"	USPAT	2003/10/27 13:57
-	3	713/201.ccls. and "adaptive security"	USPAT	2003/10/27 13:58
-	73380	security	USPAT	2003/10/27 13:58
-	0	security and "level$ or range$"	USPAT	2003/10/27 13:58
-	47499	security and (level$ or range$)	USPAT	2003/10/27 13:59
-	2	((security and (level$ or range$)) and ((adaptive or modif$ or adjust$) NEAR security)) and SOPS	USPAT	2003/10/27 14:34
-	287	(security and (level$ or range$)) and ((adaptive or modif$ or adjust$) NEAR security)	USPAT	2003/10/27 14:15
-	1	6108583.pn.	USPAT	2003/10/27 14:34
-	1	6115376.pn.	USPAT	2003/10/27 14:41
-	1	6219771.pn.	USPAT	2003/10/27 14:41
-	1	6240184.pn.	USPAT	2003/10/27 14:41
-	1	6108583.pn.	USPAT	2003/10/28 15:39
-	0	6108583.pn. and "bandwidth"	USPAT	2003/10/28 15:39
-	0	6108583.pn. and bandwidth	USPAT	2003/10/28 15:39
-	1	6510349.pn.	USPAT	2003/10/30 15:33
-	93461	bandwidth	USPAT	2003/11/03 16:53
-	127	bandwidth NEAR reallocat$	USPAT	2003/11/03 17:42
-	41	(bandwidth NEAR reallocat$) and security	USPAT	2003/11/03 18:00
-	29	(bandwidth NEAR reallocat$) and (determin$ NEAR bandwidth)	USPAT	2003/11/03 18:13
-	0	((bandwidth NEAR reallocat$) and (determin$ NEAR bandwidth)) and SOPS	USPAT	2003/11/03 18:13
-	2	((bandwidth NEAR reallocat$) and (determin$ NEAR bandwidth)) and security	USPAT	2003/11/03 18:13
-	93674	bandwidth	USPAT	2003/11/04 12:06
-	127	bandwidth NEAR reallocat$	USPAT	2003/11/04 12:13
-	109	(bandwidth NEAR reallocat$) and (security operation$)	USPAT	2003/11/04 12:13
-	108	((bandwidth NEAR reallocat$) and (security operation$)) and (transmission rate$)	USPAT	2003/11/04 12:14

-	0	(bandwidth NEAR reallocat$) and (security ADJ operation$)	USPAT	2003/11/04 12:13
-	28	((bandwidth NEAR reallocat$) and (security operation$)) and ("transmission rate$")	USPAT	2003/11/04 12:33
-	89182	(adjust$ or reallocat$ or modif$ or chang$) and bandwidth	USPAT	2003/11/04 12:36
-	2087	(adjust$ or reallocat$ or modif$ or chang$) NEAR bandwidth	USPAT	2003/11/04 12:36
-	0	((adjust$ or reallocat$ or modif$ or chang$) NEAR bandwidth) NEAR "transmission rate"	USPAT	2003/11/04 12:37
-	26	((adjust$ or reallocat$ or modif$ or chang$) NEAR bandwidth) NEAR "transmission"	USPAT	2003/11/04 12:52
-	1	((adjust$ or reallocat$ or modif$ or chang$) NEAR bandwidth) NEAR ((decreas$ or reduc$) NEAR transmission)	USPAT	2003/11/04 16:02
-	1	6115376.pn.	USPAT	2003/11/04 16:20
-	1	6219771.pn.	USPAT	2003/11/04 16:30
-	1	6240184.pn.	USPAT	2003/11/04 16:30
-	1	5793763.pn.	USPAT	2003/11/07 17:02
-	1	5689566.pn.	USPAT	2003/11/07 17:05
-	1	5477531.pn.	USPAT	2003/11/07 17:05

US005935248A

United States Patent [19]

Kuroda

[11] **Patent Number: 5,935,248**

[45] **Date of Patent: Aug. 10, 1999**

[54] **SECURITY LEVEL CONTROL APPARATUS AND METHOD FOR A NETWORK SECURING COMMUNICATIONS BETWEEN PARTIES WITHOUT PRESETTING THE SECURITY LEVEL**

[75] Inventor: **Yasutsugu Kuroda**, Kawasaki, Japan

[73] Assignee: **Fujitsu Limited**, Kawasaki, Japan

[21] Appl. No.: **08/724,757**

[22] Filed: **Oct. 3, 1996**

[30] **Foreign Application Priority Data**

Oct. 19, 1995 [JP] Japan 7-271578

[51] **Int. Cl.**^{6} **G06F 13/00**

[52] **U.S. Cl.** **713/201**

[58] **Field of Search** 395/186, 187.01, 395/200.55, 726; 364/222.5, 286.4; 340/825.31

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,012,515	4/1991	McVitie	380/49
5,018,096	5/1991	Aoyama	364/900
5,204,961	4/1993	Barlow	395/725
5,263,147	11/1993	Francisco et al.	395/425
5,355,474	10/1994	Thuraisngham et al.	395/600
5,369,707	11/1994	Follendore, III	380/25
5,475,625	12/1995	Glaschick	395/600
5,550,984	8/1996	Gelb	395/200.17
5,563,998	10/1996	Yaksich et al.	395/149
5,563,999	10/1996	Yaksich et al.	395/149
5,586,260	12/1996	Hu	395/200.2
5,596,718	1/1997	Boebert et al.	395/187.01
5,675,782	10/1997	Montague et al.	395/609
5,699,513	12/1997	Feigen et al.	395/187.01

FOREIGN PATENT DOCUMENTS

0375138 A2	6/1990	European Pat. Off.	H04L 12/58
0409397 A2	6/1990	European Pat. Off.	G06F 1/00
0375139 A2	1/1991	European Pat. Off.	H04L 12/58
0520709 A2	12/1992	European Pat. Off.	G06F 1/00
0534679 A2	3/1993	European Pat. Off.	G07C 9/00

OTHER PUBLICATIONS

Varadharajan et al., "A Multilevel Security Model for a Distributed Object–Oriented System", IEEE, pp. 68–78, Dec. 1990.

Hinke, "The Trusted Server Approach to Multilevel Security" IEEE, pp. 335–341, Dec. 1989.

Primary Examiner—Joseph E. Palys
Assistant Examiner—Stephen C. Elmore
Attorney, Agent, or Firm—Staas & Halsey LLP

[57] **ABSTRACT**

In a security level control apparatus for controlling a security level of a communication established between communication parties, this security level control apparatus is arranged by employing a security level recognizing unit and a security level setting unit. The security level recognizing unit recognizes a security level notified from a communication party. The security level setting unit sets the security level recognized by the security level recognizing unit as a security level for the security level control apparatus. In accordance with this security level control apparatus, the security level of the communication party recognized by the security level recognizing unit is first set as the security level for the security level control apparatus. As a result, the communication can be established between the communication parties without presetting the security level.

13 Claims, 13 Drawing Sheets

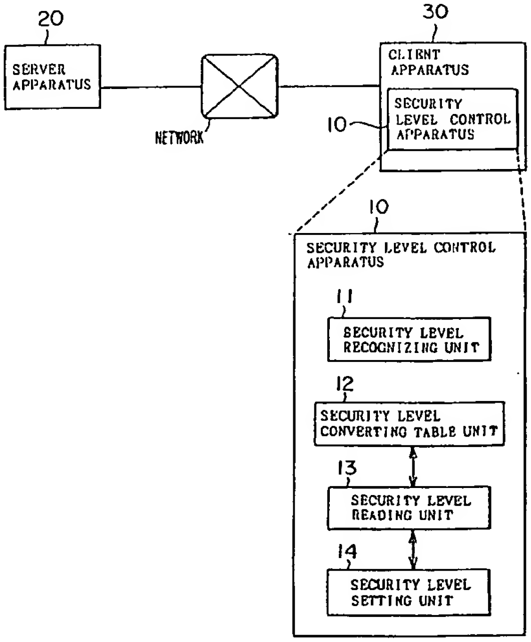

20
SERVER APPARATUS
NETWORK
30
CLIENT APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
14
SECURITY LEVEL SETTING UNIT

FIG. 1

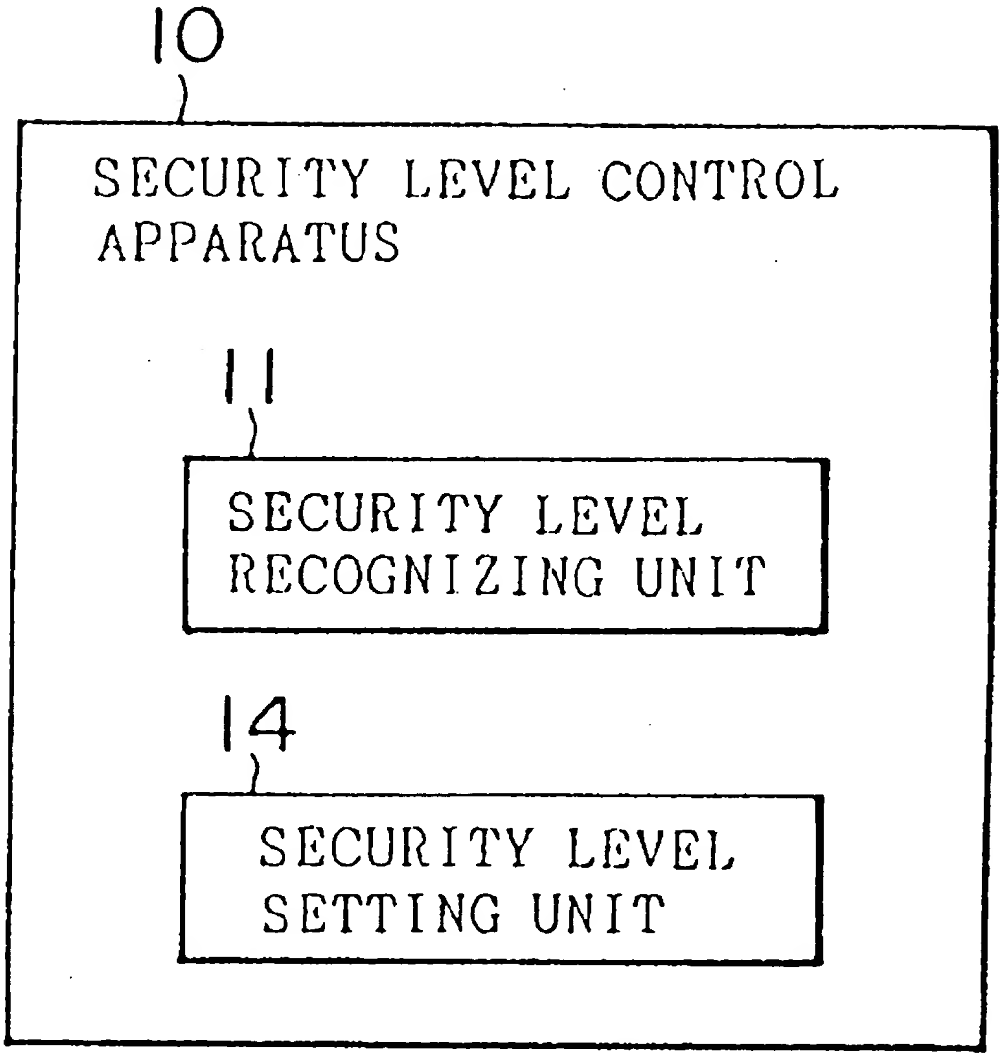

10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
14
SECURITY LEVEL SETTING UNIT

FIG. 2

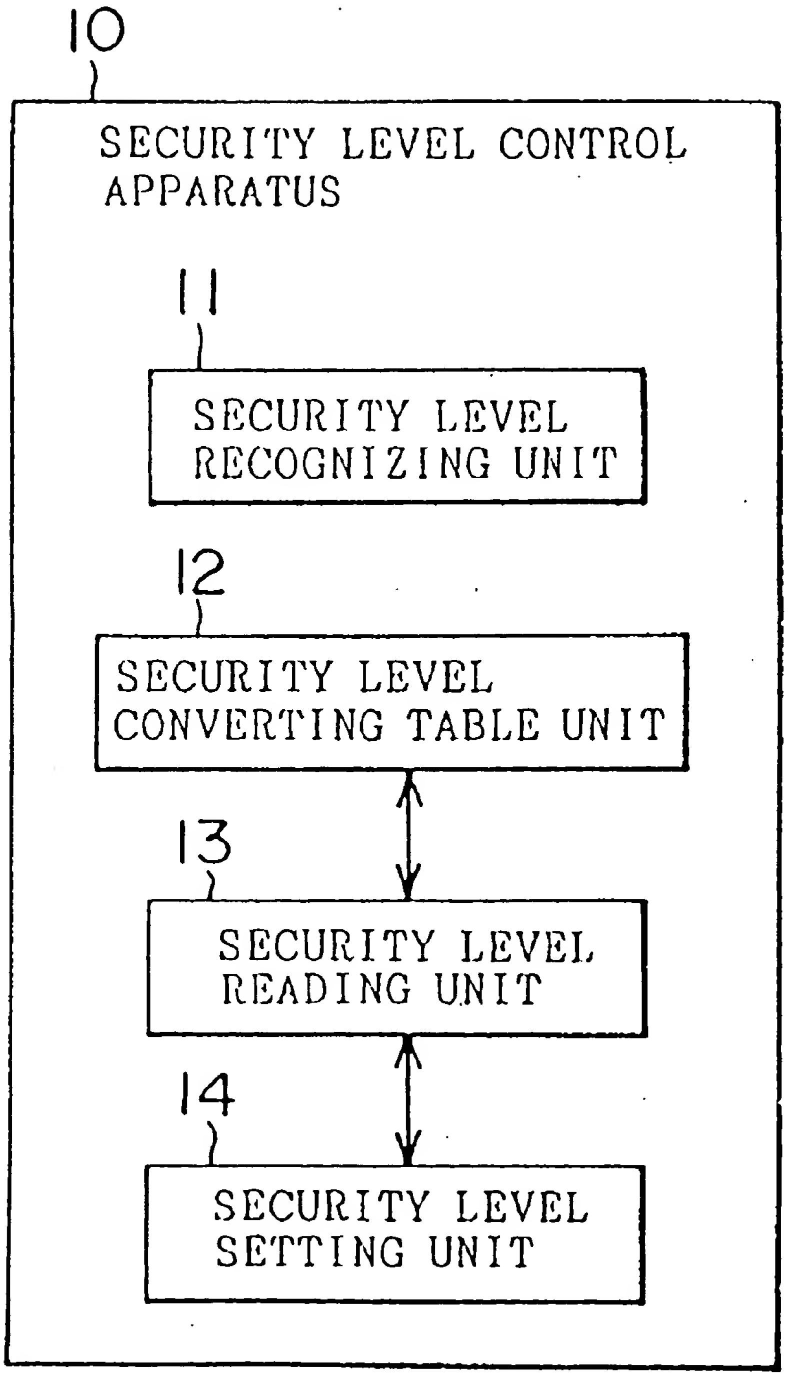

10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
14
SECURITY LEVEL SETTING UNIT

FIG. 3

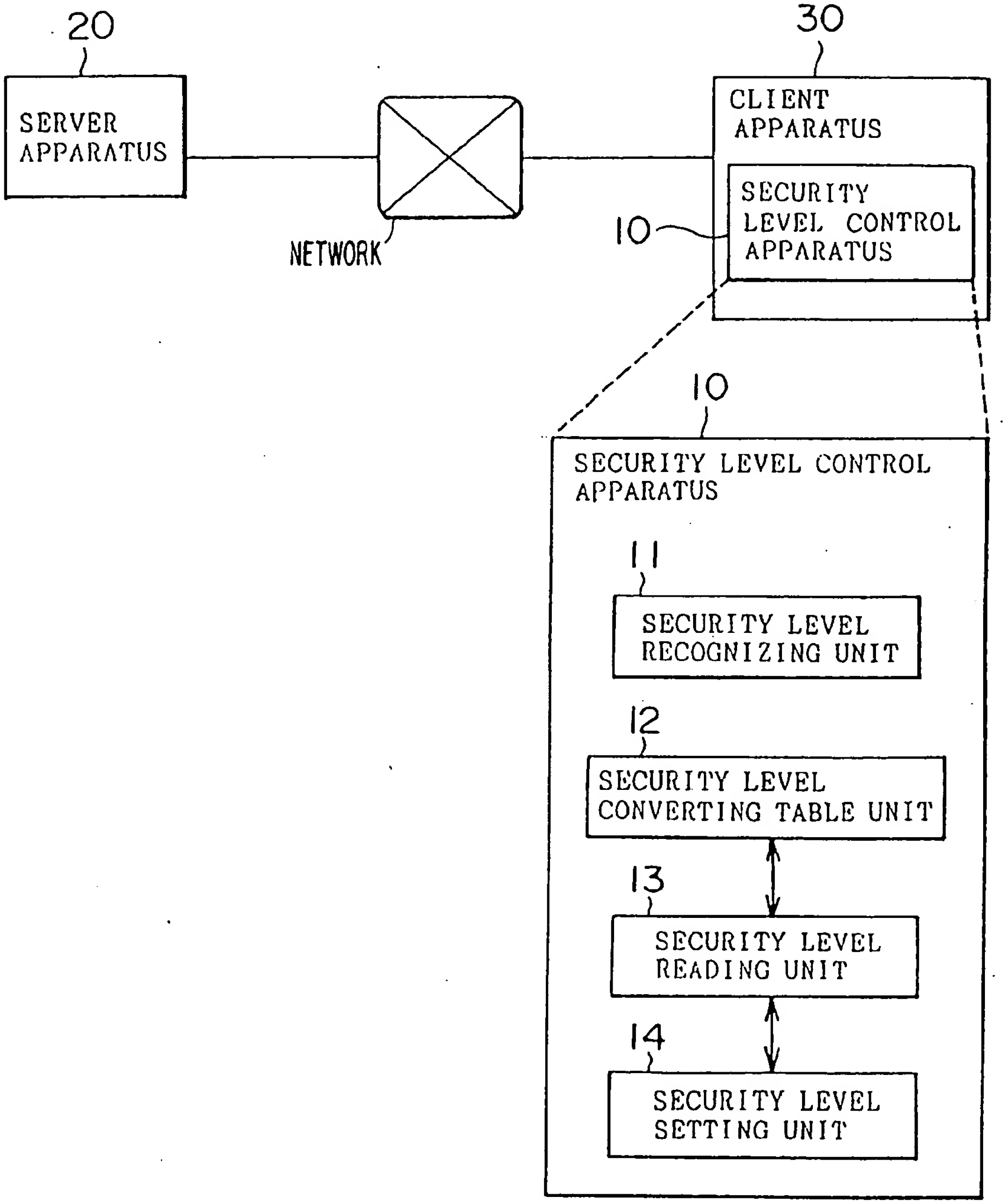

20
SERVER APPARATUS
NETWORK
30
CLIENT APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
14
SECURITY LEVEL SETTING UNIT

FIG. 4

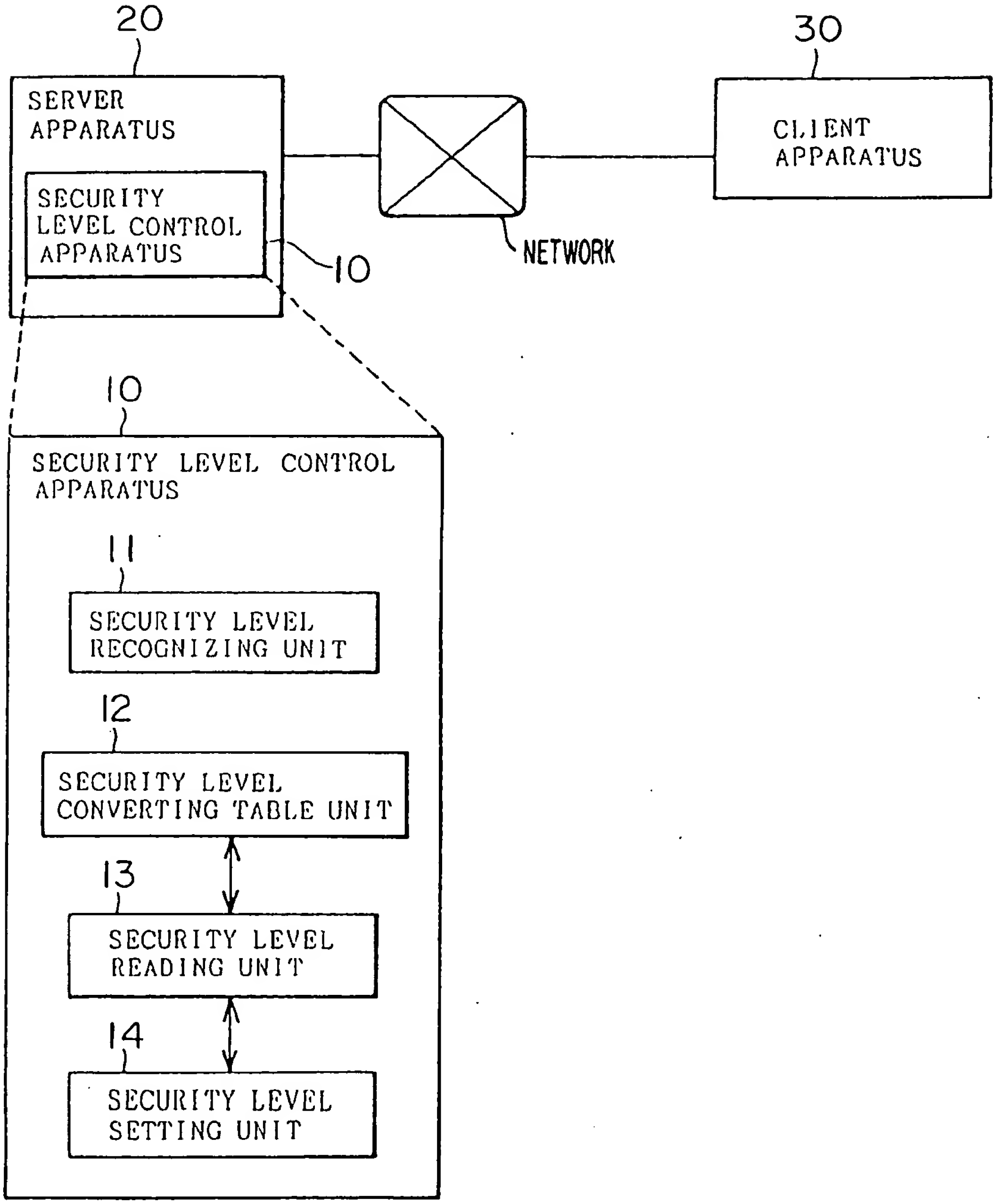

20
SERVER APPARATUS
SECURITY LEVEL CONTROL APPARATUS
10
NETWORK
30
CLIENT APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
14
SECURITY LEVEL SETTING UNIT

FIG. 5

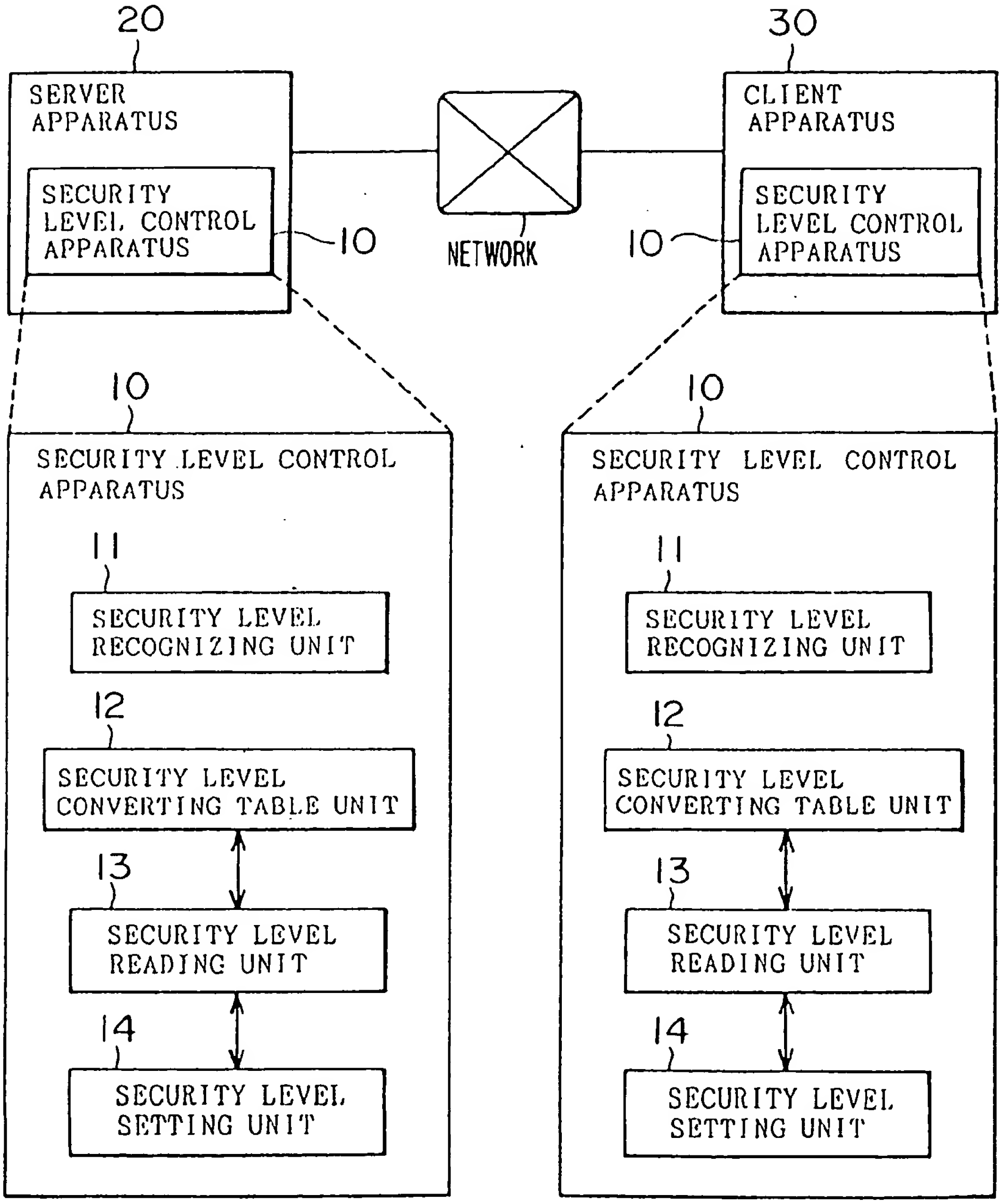

20
SERVER APPARATUS
SECURITY LEVEL CONTROL APPARATUS
10
NETWORK
30
CLIENT APPARATUS
SECURITY LEVEL CONTROL APPARATUS
10
10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
14
SECURITY LEVEL SETTING UNIT
10
SECURITY LEVEL CONTROL APPARATUS
11
SECURITY LEVEL RECOGNIZING UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
14
SECURITY LEVEL SETTING UNIT

FIG. 6

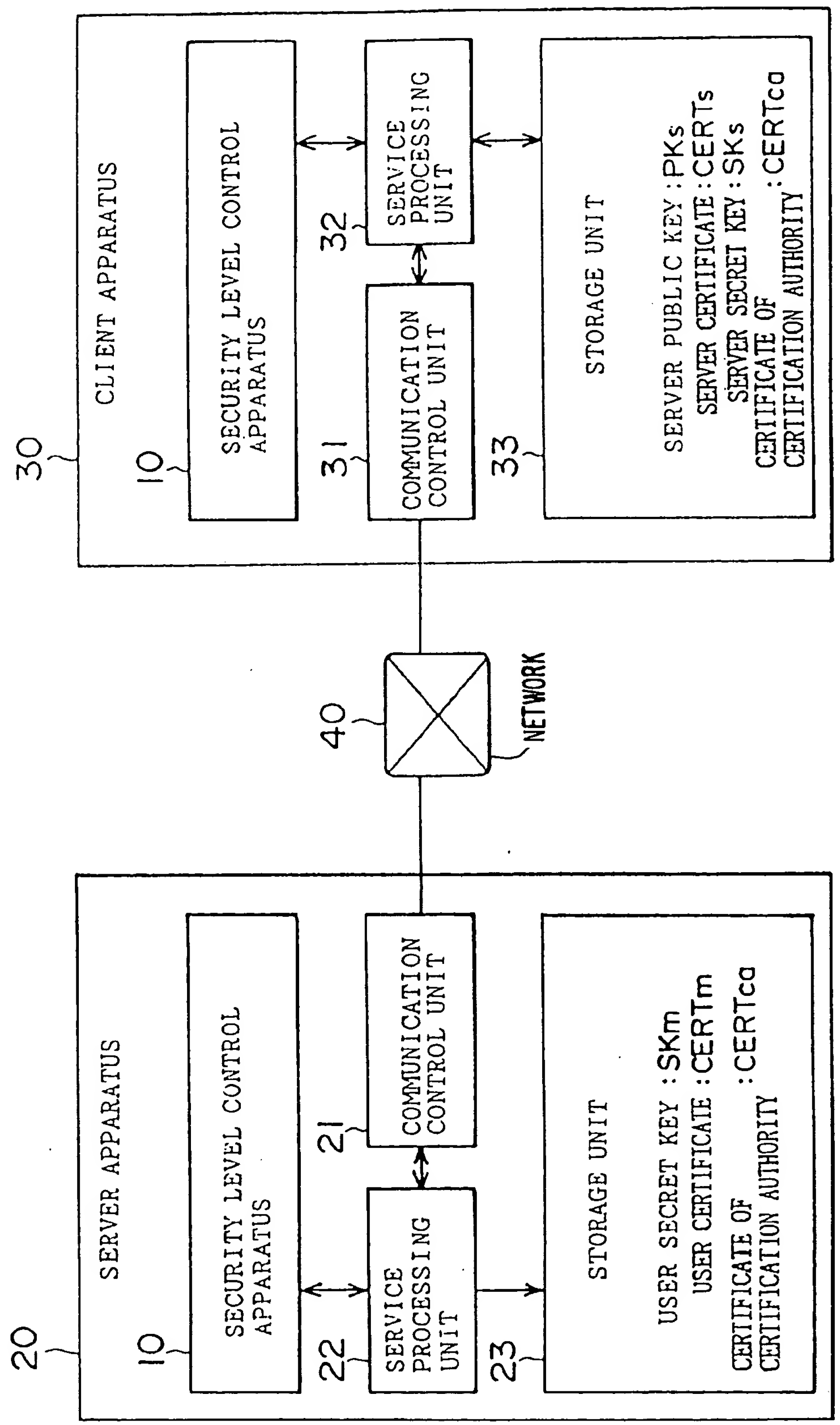

30
CLIENT APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
32
SERVICE PROCESSING UNIT
31
COMMUNICATION CONTROL UNIT
33
STORAGE UNIT
SERVER PUBLIC KEY : PKs
SERVER CERTIFICATE : CERTs
SERVER SECRET KEY : SKs
CERTIFICATE OF CERTIFICATION AUTHORITY :CERTca
40
NETWORK
20
SERVER APPARATUS
10
SECURITY LEVEL CONTROL APPARATUS
21
COMMUNICATION CONTROL UNIT
22
SERVICE PROCESSING UNIT
23
STORAGE UNIT
USER SECRET KEY : SKm
USER CERTIFICATE : CERTm
CERTIFICATE OF CERTIFICATION AUTHORITY :CERTca

FIG. 7

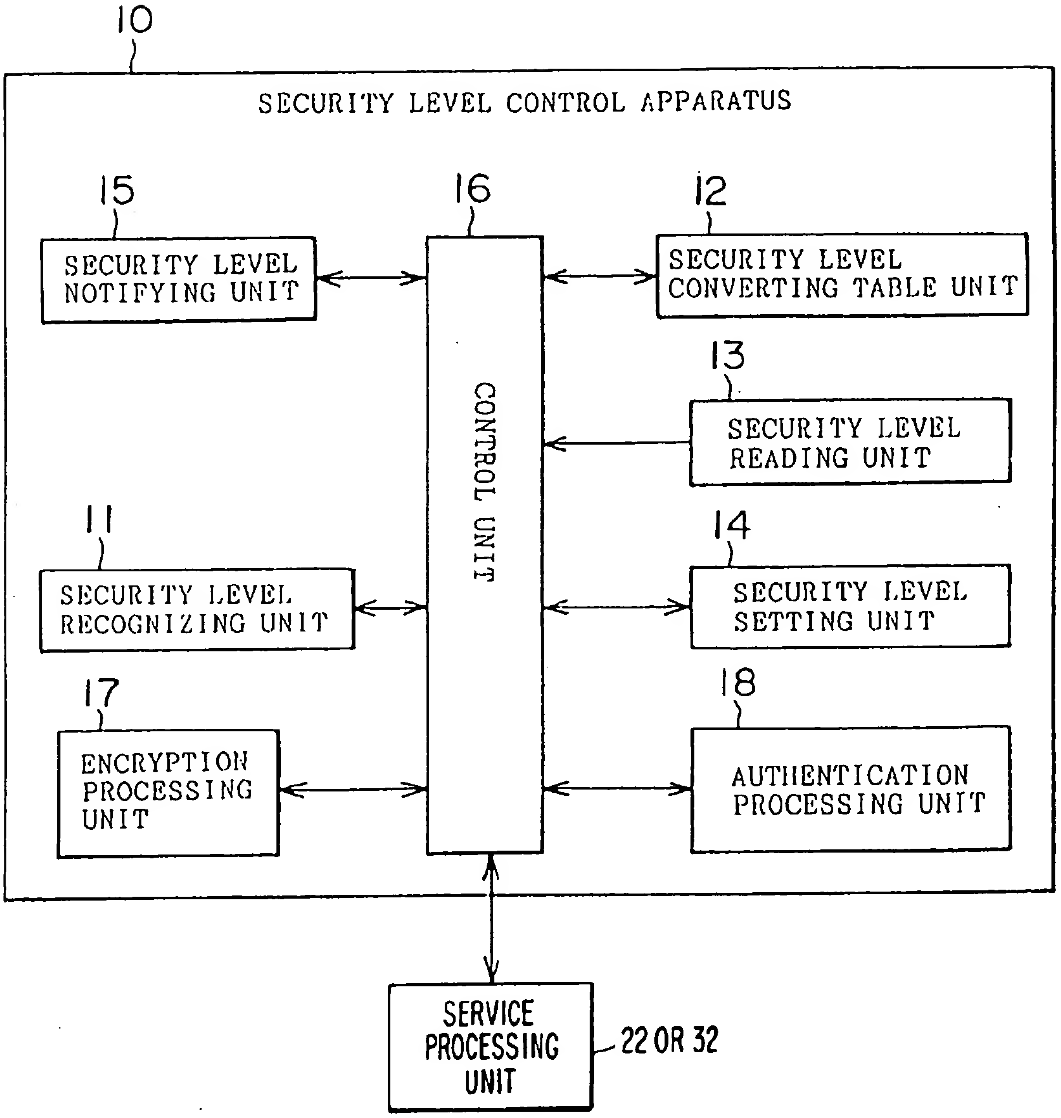

10
SECURITY LEVEL CONTROL APPARATUS
15
SECURITY LEVEL NOTIFYING UNIT
16
CONTROL UNIT
12
SECURITY LEVEL CONVERTING TABLE UNIT
13
SECURITY LEVEL READING UNIT
11
SECURITY LEVEL RECOGNIZING UNIT
14
SECURITY LEVEL SETTING UNIT
17
ENCRYPTION PROCESSING UNIT
18
AUTHENTICATION PROCESSING UNIT
SERVICE PROCESSING UNIT
22 OR 32

FIG. 8

SECURITY LEVEL OF SERVER SECURITY LEVEL OF CLIENT	1	2	3	4	5
1	1	2	×	4	×
2	2	2	×	4	×
3	×	2	3	×	5
4	×	×	×	4	×
5	×	×	×	4	5

FIG. 9

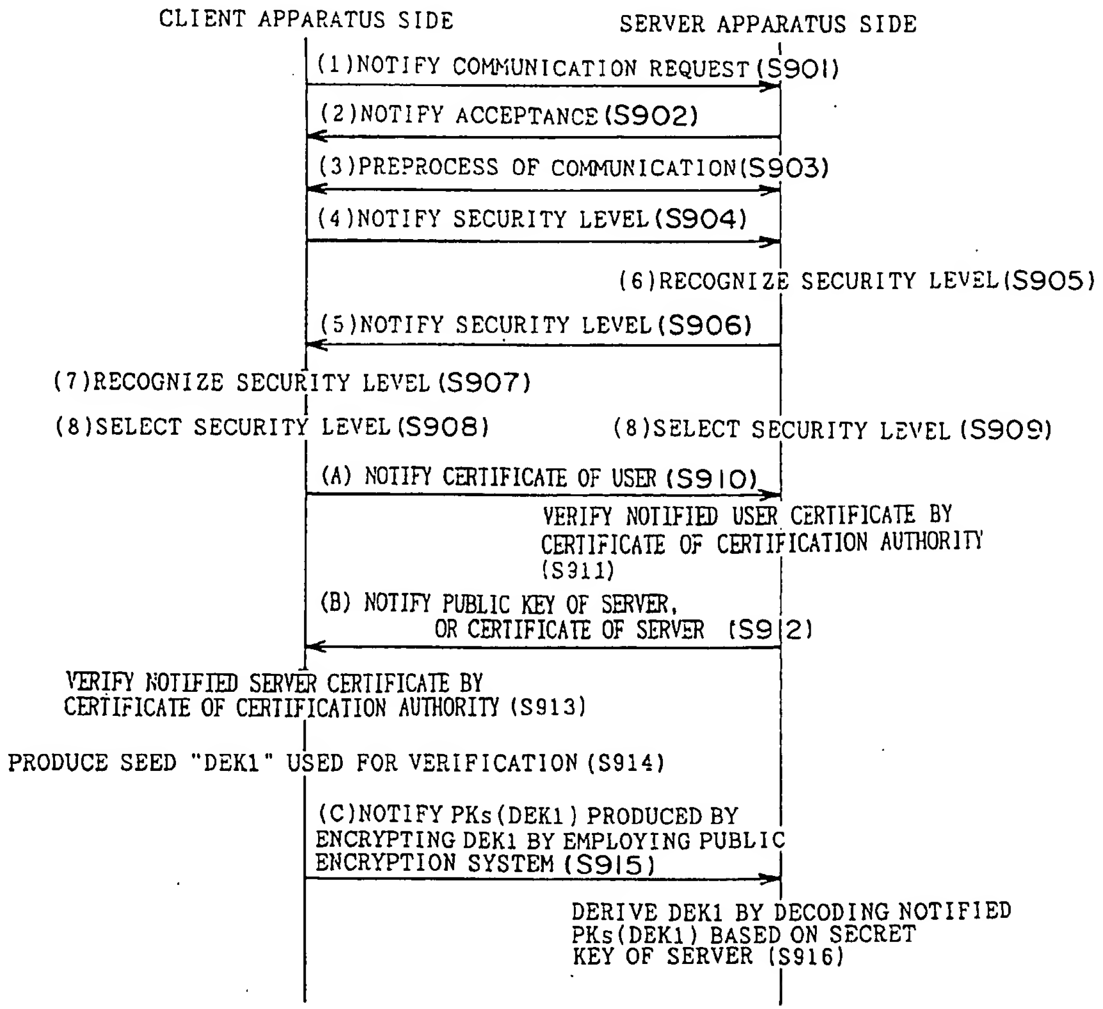

CLIENT APPARATUS SIDE
SERVER APPARATUS SIDE
(1)NOTIFY COMMUNICATION REQUEST(S901)
(2)NOTIFY ACCEPTANCE(S902)
(3)PREPROCESS OF COMMUNICATION(S903)
(4)NOTIFY SECURITY LEVEL(S904)
(6)RECOGNIZE SECURITY LEVEL(S905)
(5)NOTIFY SECURITY LEVEL(S906)
(7)RECOGNIZE SECURITY LEVEL (S907)
(8)SELECT SECURITY LEVEL(S908)
(8)SELECT SECURITY LEVEL (S909)
(A) NOTIFY CERTIFICATE OF USER (S910)
VERIFY NOTIFIED USER CERTIFICATE BY CERTIFICATE OF CERTIFICATION AUTHORITY (S911)
(B) NOTIFY PUBLIC KEY OF SERVER, OR CERTIFICATE OF SERVER (S912)
VERIFY NOTIFIED SERVER CERTIFICATE BY CERTIFICATE OF CERTIFICATION AUTHORITY (S913)
PRODUCE SEED "DEK1" USED FOR VERIFICATION (S914)
(C)NOTIFY PKs(DEK1) PRODUCED BY ENCRYPTING DEK1 BY EMPLOYING PUBLIC ENCRYPTION SYSTEM (S915)
DERIVE DEK1 BY DECODING NOTIFIED PKs(DEK1) BASED ON SECRET KEY OF SERVER (S916)

FIG. 10

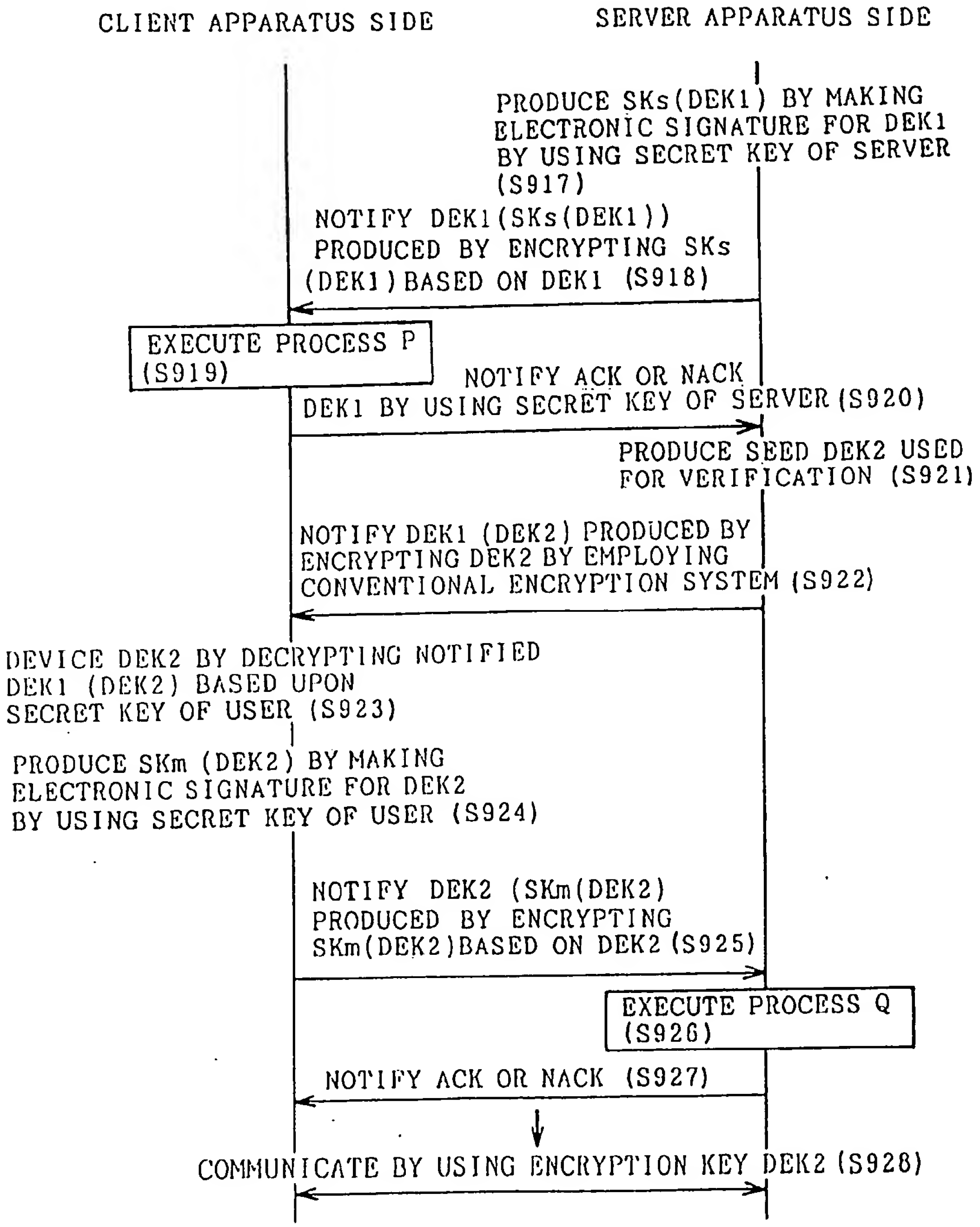

CLIENT APPARATUS SIDE
SERVER APPARATUS SIDE
PRODUCE SKs(DEK1) BY MAKING ELECTRONIC SIGNATURE FOR DEK1 BY USING SECRET KEY OF SERVER (S917)
NOTIFY DEK1(SKs(DEK1)) PRODUCED BY ENCRYPTING SKs (DEK1) BASED ON DEK1 (S918)
EXECUTE PROCESS P (S919)
NOTIFY ACK OR NACK DEK1 BY USING SECRET KEY OF SERVER (S920)
PRODUCE SEED DEK2 USED FOR VERIFICATION (S921)
NOTIFY DEK1 (DEK2) PRODUCED BY ENCRYPTING DEK2 BY EMPLOYING CONVENTIONAL ENCRYPTION SYSTEM (S922)
DEVICE DEK2 BY DECRYPTING NOTIFIED DEK1 (DEK2) BASED UPON SECRET KEY OF USER (S923)
PRODUCE SKm (DEK2) BY MAKING ELECTRONIC SIGNATURE FOR DEK2 BY USING SECRET KEY OF USER (S924)
NOTIFY DEK2 (SKm(DEK2) PRODUCED BY ENCRYPTING SKm(DEK2)BASED ON DEK2 (S925)
EXECUTE PROCESS Q (S926)
NOTIFY ACK OR NACK (S927)
COMMUNICATE BY USING ENCRYPTION KEY DEK2 (S928)

FIG. 11

SECURITY LEVEL "1" (1)→(2)→(3)→(4)→(5)→(6)→(7)→(8)→(9)

SECURITY LEVEL "2" (1)→(2)→(3)→(4)→(5)→(6)→(7)→(8)→(B)→(C)→(F)

SECURITY LEVEL "3" (1)→(2)→(3)→(4)→(5)→(6)→(7)→(8)→(A)→(B)→(C)→(F)→(G)→(H)

SECURITY LEVEL "4" (1)→(2)→(3)→(4)→(5)→(6)→(7)→(8)→(B)→(C)→(D)→(E)→(F)

SECURITY LEVEL "5" (1)→(2)→(3)→(4)→(5)→(6)→(7)→(8)→(A)→(B)→(C)→(D)→(E)→(F)→(G)→(H)

FIG. 12

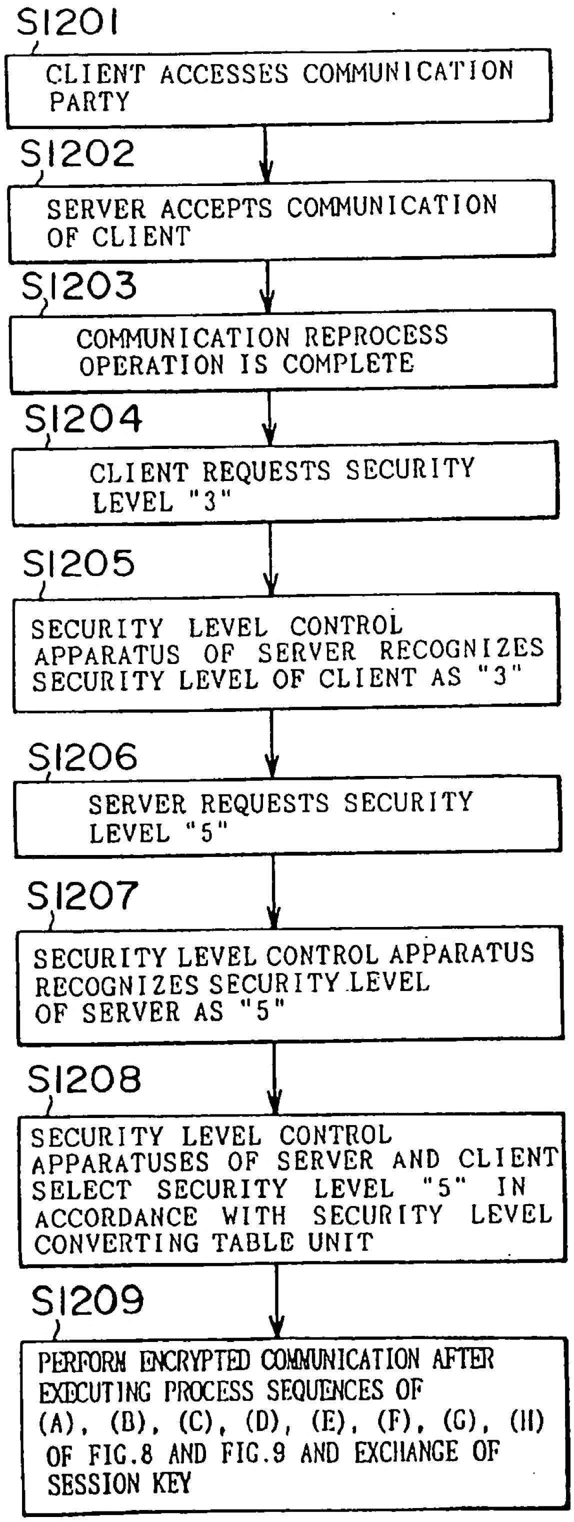
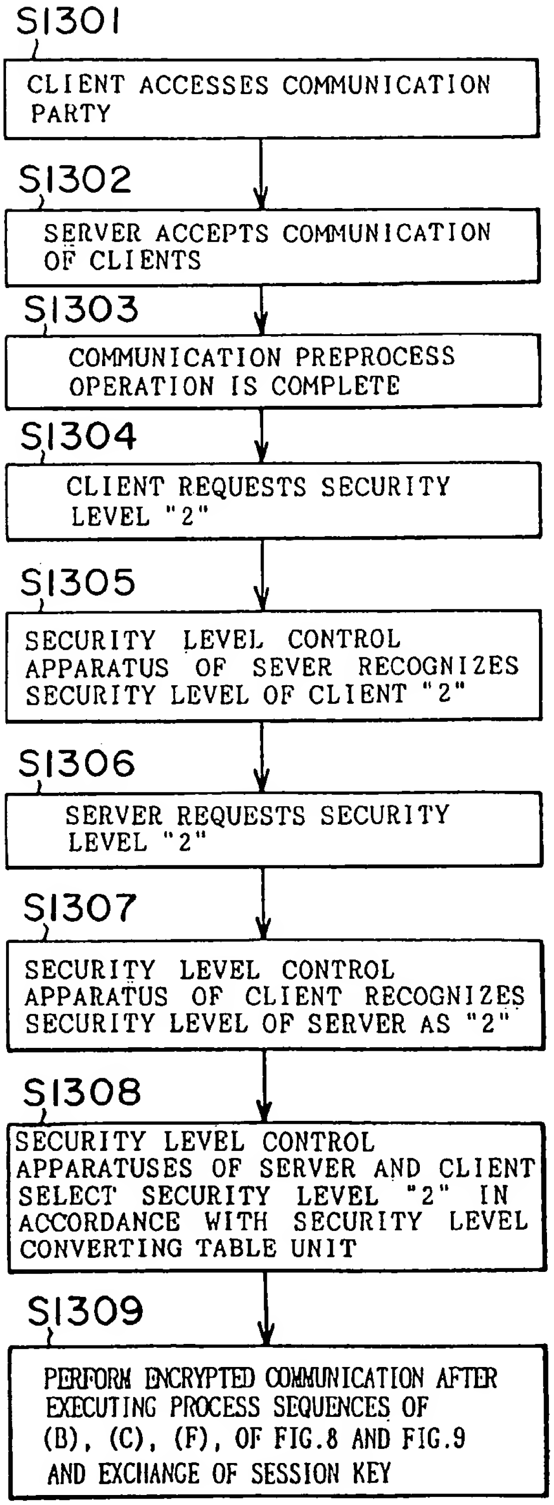

S1201
CLIENT ACCESSES COMMUNICATION PARTY
S1202
SERVER ACCEPTS COMMUNICATION OF CLIENT
S1203
COMMUNICATION REPROCESS OPERATION IS COMPLETE
S1204
CLIENT REQUESTS SECURITY LEVEL "3"
S1205
SECURITY LEVEL CONTROL APPARATUS OF SERVER RECOGNIZES SECURITY LEVEL OF CLIENT AS "3"
S1206
SERVER REQUESTS SECURITY LEVEL "5"
S1207
SECURITY LEVEL CONTROL APPARATUS RECOGNIZES SECURITY LEVEL OF SERVER AS "5"
S1208
SECURITY LEVEL CONTROL APPARATUSES OF SERVER AND CLIENT SELECT SECURITY LEVEL "5" IN ACCORDANCE WITH SECURITY LEVEL CONVERTING TABLE UNIT
S1209
PERFORM ENCRYPTED COMMUNICATION AFTER EXECUTING PROCESS SEQUENCES OF (A), (B), (C), (D), (E), (F), (G), (H) OF FIG.8 AND FIG.9 AND EXCHANGE OF SESSION KEY

FIG. 13


S1301
CLIENT ACCESSES COMMUNICATION PARTY
S1302
SERVER ACCEPTS COMMUNICATION OF CLIENTS
S1303
COMMUNICATION PREPROCESS OPERATION IS COMPLETE
S1304
CLIENT REQUESTS SECURITY LEVEL "2"
S1305
SECURITY LEVEL CONTROL APPARATUS OF SEVER RECOGNIZES SECURITY LEVEL OF CLIENT "2"
S1306
SERVER REQUESTS SECURITY LEVEL "2"
S1307
SECURITY LEVEL CONTROL APPARATUS OF CLIENT RECOGNIZES SECURITY LEVEL OF SERVER AS "2"
S1308
SECURITY LEVEL CONTROL APPARATUSES OF SERVER AND CLIENT SELECT SECURITY LEVEL "2" IN ACCORDANCE WITH SECURITY LEVEL CONVERTING TABLE UNIT
S1309
PERFORM ENCRYPTED COMMUNICATION AFTER EXECUTING PROCESS SEQUENCES OF (B), (C), (F), OF FIG.8 AND FIG.9 AND EXCHANGE OF SESSION KEY

SECURITY LEVEL CONTROL APPARATUS AND METHOD FOR A NETWORK SECURING COMMUNICATIONS BETWEEN PARTIES WITHOUT PRESETTING THE SECURITY LEVEL

BACKGROUND OF THE INVENTION

The present invention relates to a security level control apparatus, and more specifically, to a security level control apparatus for controlling security levels of communications established between communication parties.

Also, the present invention relates to a network communication system, and in particular, to a network communication system constituted by a server apparatus and a client apparatus, which perform communications, the security levels of which are set.

Network services through which electronic mails are provided are commercially available by mutually connecting computers installed in a distribution manner.

However, in network systems configured for academic purposes, typically known as the Internet, proper care is not taken to network security matters. Accordingly, these network services involve various problems, for instance wiretapping, falsification, and impersonation.

Now, a description will be made of these wiretapping, falsification, and impersonation with respect to electronic mails.

The term "wiretapping" implies that a plain text, i.e., a correspondence message not yet encrypted is read during message transmission.

The term "falsification" implies that a content of an electronic mail is modified. This falsification is performed in relaying nodes when an electronic mail is delivered via a plurality of relaying nodes.

The term "impersonation" implies that when no protection is established with respect to information for specifying a mail sender, a third party (bearing offense) falsifies the information for specifying the third party to pose as an impersonator.

To solve these network problems, at least one of the following solutions is carried out as follows. For instance, a message (data) is encrypted, an electronic signature (Message Integrity Check) is used to prevent falsification, and a user (communication party) is authenticated. In such a network communication system realized in a server/client manner, the server apparatus is authenticated and/or the client apparatus is authenticated.

As to encryption techniques, the secret key cryptosystem, the public key cryptosystem, and the like are known. In the secret key cryptosystem, the encrypting operation and the decrypting operation are carried out by using the common key between the communication parties. On the other hand, in the public key encrypting system, the key system is constituted by combining the secret keys for the individual users with the public keys, and the public keys are opened to the third party, whereas the secret keys are disclosed only to the individual users. In this public key cryptosystem, a message which has been encrypted by the public key can be solved by the secret key. For instance, when a message is transmitted from "A" to "B", "A" encrypts this message by using the public key of "B", and then "B" who has received the encrypted message can decrypt this encrypted message by using the own secret key. The person who can decrypt this encrypted message is only "B" who knows the own secret key.

As to the authentication techniques, the password authentication and the electronic signature with employment of the public key cryptosystem are known.

In the above-described conventional network techniques, a plurality of security levels are produced when certain process operations are combined with each other in order to avoid the problems such as wiretapping, falsification, and impersonation with respect to the network services.

For instance, it is conceivable that a resultant security level becomes high when electronic mail is encrypted and at the same time a user of this electronic mail is authenticated, rather than only the encryption of this electronic mail. When only the security should be emphasized, it is best to combine a large number of processing operations with each other. However, in this case, the resultant workloads would be increased.

Under such a circumstance, it is a proper solution to set the security level to which importance of a communication content is reflected. Proper setting of a security level based on importance of a message is called a "policy of security".

With respect to this "policy of security", the below-mentioned problems occur in the above-described conventional techniques.

That is, as to the first problem, the communication is performed between the communication parties in accordance with only a predetermined security policy, but cannot be carried out in accordance with other security policies.

As to the second problem, the security level of the communication party (communication destination) is continuously introduced with a top priority, so that the security level cannot be determined.

The present invention has been made to solve the above-described problems, and therefore, has a first object to provide a security level control apparatus and a network communication system, capable of executing a communication between communication parties, while a security level is not previously determined.

Also, a second object of the present invention is to provide a security level control apparatus and a network communication system, capable of executing a communication while determining the own security level.

SUMMARY OF THE INVENTION

To achieve the above-described objects, a security level control apparatus of the present invention is featured as follows. In a security level control apparatus for controlling a security level of a communication established between communication parties, this security level control apparatus is arranged by employing a security level recognizing unit and a security level setting unit.

The security level recognizing unit recognizes a security level notified from a communication party.

The security level setting unit sets the security level recognized by the security level recognizing unit as a security level for the security level control apparatus. In accordance with this security level control apparatus, the security level of the communication party recognized by the security level recognizing unit is first set as the security level for the security level control apparatus. As a result, the communication can be established between the communication parties without presetting the security level.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the teachings of the present invention may be acquired by referring to the accompanying drawings, in which:

FIG. 1 is a block diagram for schematically showing a basic idea of a first security level control apparatus according to the present invention;

FIG. 2 is a block diagram for schematically showing a basic idea of a second security level control apparatus according to the present invention;

FIG. 3 is a block diagram for schematically indicating a basic idea of a first network communication system according to the present invention;

FIG. 4 is a block diagram for schematically indicating a basic idea of a second network communication system according to the present invention;

FIG. 5 is a block diagram for schematically indicating a basic idea of a ninth network communication system according to the present invention;

FIG. 6 schematically represents an arrangement of a system according to an embodiment of the present invention;

FIG. 7 is a schematic block diagram for showing an arrangement of the security level control apparatus of the embodiment;

FIG. 8 shows a security level conversion table unit included by the security level control apparatus of the embodiment;

FIG. 9 schematically represents a first sequential process operation executed between the client apparatus and the server apparatus according to the embodiment;

FIG. 10 schematically represents a second sequential process operation executed between the client apparatus and the server apparatus according to the embodiment;

FIG. 11 schematically indicates a sequential process operation executed in the respective security levels of the embodiment;

FIG. 12 is a flow chart for indicating a first process operation according to the embodiment; and

FIG. 13 is a flow chart for indicating a second process operation according to the embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENT

First Security Level Control Apparatus 10

To solve the above-described first problem, a first security level control apparatus **10** of the present invention is arranged as follows (corresponding to claim **1**). FIG. **1** is a schematic block diagram for showing a basic idea of the security level control apparatus **10** according to the present invention.

That is, the security level control apparatus **10** for controlling a security level of a communication executed between communication parties is arranged by a security level recognizing unit **11** and a security level setting unit **14**.

(Security Level Recognizing Unit **11**)

The security level recognizing apparatus **11** may recognize a security level notified from communication destination (communication party).

(Security Level Setting Unit **14**)

The security level setting unit **14** may set the security level recognized by the security level recognizing unit **11** as a security level for the security level control apparatus **10**.

In accordance with the first security level control apparatus **10** of the present invention, the following operations are carried out. First, the security level of the communication party recognized by the security level recognizing unit **11** is set as the security level for the security level control apparatus **10**.

Second Security Level Control Apparatus 10

To solve the above-described second problem, a second security level control apparatus **10** of the present invention is arranged as follows (corresponding to claim **2**). FIG. **1** is a schematic block diagram for showing a basic idea of the security level control apparatus **10** according to the present invention.

That is, the security level control apparatus **10** for controlling a security level of a communication executed between communication parties is arranged by a security level recognizing unit **11**, a security level converting table unit **12**, a security level reading unit **13**, and a security level setting unit **14**.

(Security Level Recognizing Unit **11**)

The security level recognizing apparatus **11** may recognize a security level notified from communication destination (communication party).

(Security Level Converting Table Unit **12**)

The security level converting table unit **12** may store a relationship between an index made of two sets of security levels, and a security level of an actual communication.

(Security Level Reading Unit **13**)

The security level reading unit **13** may read from the security level converting table unit **12**, a security level corresponding to such an index. This index is defined by the security level of the communication party recognized by the security level recognizing unit **11**, and the security level for the security level control apparatus **10**.

(Security Level Setting Unit **14**)

The security level setting unit **14** may set the security level recognized by the security level recognizing unit **11** as the security level for the security level control apparatus **10**.

In accordance with the second security level control apparatus **10** of the present invention, the following operations are carried out. First, both the security level of the communication party recognized by the security level recognizing unit **11** and the security level for the security level control apparatus **10** are set as the index. Then, the security level corresponding to this index is read from the security level converting table unit **12**. Thus, this read security level is set as the security level for the security level apparatus **10**.

First Network Communication System

A first network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described first problem (corresponding to claim **3**). FIG. **3** is a schematic block diagram for indicating a basic idea corresponding to the first network communication system of the present invention.

That is, in a network communication system provided with a server apparatus **20** and a client apparatus **30**, which perform a communication whose security level is set, the client apparatus **30** includes a security level control apparatus **10**. Then, this security level control apparatus **10** is constructed of a security level recognizing unit **11** and a security level setting unit **14**.

(Security Level Recognizing Unit **11**)

The security level recognizing unit **11** may recognize a security level notified from a communication party (communication destination).

(Security Level Setting Unit **14**)

The security level setting unit **14** may set the security level recognized by the security level recognizing unit **11** as a security level for the client apparatus **30**.

In accordance with the first network communication system of the present invention, the following operations are carried out on the side of the client apparatus 30. First, the security level of the server apparatus 20 recognized by the security level recognizing unit 11 is set as the security level for the client apparatus 30.

Second Network Communication System

A second network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described first problem (corresponding to claim 4). FIG. 4 is a schematic block diagram for indicating a basic idea corresponding to the second network communication system of the present invention.

That is, in a network communication system provided with a server apparatus 20 and a client apparatus 30, which perform a communication whose security level is set, the server apparatus 20 includes a security level control apparatus 10. Then, this security level control apparatus 10 is constructed of a security level recognizing unit 11 and a security level setting unit 14.

(Security Level Recognizing Unit 11)

The security level recognizing unit 11 may recognize a security level notified from a communication party (communication destination).

(Security Level Setting Unit 14)

The security level setting unit 14 may set the security level recognized by the security level recognizing unit 11 as a security level for the server apparatus 20.

In accordance with the second network communication system of the present invention, the following operations are carried out on the side of the server apparatus 20. First, the security level of the client apparatus 30 recognized by the security level recognizing unit 11 is set as the security level for the server apparatus 20.

Third Network Communication System

A third network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described first problem (corresponding to claim 5).

That is, in the first network communication system, a plurality of the server apparatus 20 are employed.

Then, the security level control apparatus 10 provided with the client apparatus 30 may control the security level with respect to each of the server apparatus 20.

In accordance with the third network communication system of the present invention, the following operations are carried out. That is, the security levels are controlled by the server apparatus 20.

Fourth Network Communication System

A fourth network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described first problem (corresponding to claim 6). That is, in the second network communication system, a plurality of the above-described client apparatuses 30 are employed. Then, the security level control apparatus 10 employed in the server apparatus 20 may control the security levels with respect to each of the client apparatuses 30.

According to the fourth network communication system, the following operation is performed. That is, the security level is controlled with respect to each of the client apparatuses 30.

Fifth Network Communication System

A fifth network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described second problem (corresponding to claim 7).

That is, in either the first or third network communication system, the security level control apparatus 10 owned by the client apparatus 30 includes a security level converting table unit 12 and a security level reading unit 13.

The security level converting table unit 12 stores a relationship between an index constructed of two sets of security levels and an actual communication security level.

The security level reading unit 13 reads from the security level converting table unit 12, a security level corresponding to such an index that is constructed of a security level of a communication party recognized by the security level recognizing unit 11, and the security level for the client apparatus 30.

Then, the security level setting unit 14 sets the security level read from the security level reading unit 13 as the security level for the client apparatus 30.

In accordance with the fifth network communication system of the present invention, the following operations are carried out. First, the security level of the server apparatus 20 recognized by the security level recognizing unit 11, and the security level for the client apparatus 30 are used as the index. Then, the security level corresponding to this index is read from the security level converting table unit 12. This read security level is set as a security level for the client apparatus 30.

Sixth Network Communication System

A sixth network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described second problem (corresponding to claim 8).

That is, in either the second or fourth network communication system, the security level control apparatus 10 owned by the server apparatus 20 includes a security level converting table unit 12 and a security level reading unit 13.

The security level converting table unit 12 stores a relationship between an index constructed of two sets of security levels and an actual communication security level.

The security level reading unit 13 reads from the security level converting table unit 12, a security level corresponding to such an index that is constructed of a security level of the client apparatus 30 recognized by the security level recognizing unit 11 and the security level for the server apparatus 20.

Then, the security level setting unit 14 sets the security level read from the security level reading unit 13 as the security level for the server apparatus 20. In accordance with the sixth network communication system of the present invention, the following operations are carried out. First, the security level of the client apparatus 30 recognized by the security level recognizing unit 11, and the security level for the server apparatus 20 are used as the index. Then, the security level corresponding to this index is read from the security level converting table unit 12. This read security level is set as a security level for the server apparatus 20.

Seventh Network Communication System

A seventh network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described second problem (corresponding to claim 9).

That is, in the fifth network communication system, the security level control apparatus 10 owned by the client apparatus 30 may dynamically change the security level even during the communication in response to a request from the client apparatus 30.

In accordance with the seventh network communication system of the present invention, the following operations are carried out. That is, the security levels are dynamically variable even during the communication.

Eighth Network Communication System

An eighth network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described second problem (corresponding to claim **10**).

That is, in the sixth network communication system, the security level control apparatus owned by the server apparatus may dynamically change the security level even during the communication in response to a request from the server apparatus.

In accordance with the eighth network communication system of the present invention, the following operations are carried out. That is, the security levels are dynamically variable even during the communication.

Ninth Network Communication System

A ninth network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described second problem (corresponding to claim **11**). FIG. **5** is a schematic block diagram for indicating a basic idea corresponding to the ninth network communication system of the present invention.

That is, in a network communication system provided with the server apparatus **20** and the client apparatus **30**, which perform a communication whose security level is set, the server apparatus **20** and the client apparatus **30** include security level control apparatuses **10**. Then, this security level control apparatus **10** is constructed of a security level recognizing unit 11, a security level converting table unit **12**, a security level reading unit **13**, and a security level setting unit **14**.

(Security-Level Recognizing Unit 11)

The security level recognizing unit 11 may recognize a security level notified from a communication party.

(Security Level Converting Table Unit **12**)

The security level converting table unit **12** stores a relationship between an index constructed of two sets of security levels and an actual communication security level.

(Security Level Reading Unit **13**)

The security level reading unit **13** reads from the security level converting table unit **12**, a security level corresponding to such an index that is constructed of the security level of the communication party recognized by the security level recognizing unit 11 and the security level for the security level control apparatus **10**.

(Security Level Setting Unit **14**)

The security level setting unit **14** sets the security level read from the security level reading unit **13** as the security level for the security level control apparatus **10**.

In accordance with the ninth network communication system, the following operations are carried out. First, both the security level of the communication party recognized by the security level recognizing unit 11 and the security level for the security level control apparatus **10** are used as the index. Then, the security level corresponding to this index is read from the security level converting table unit **12**. This read security level is set as the security level for the security level apparatus **10**.

Tenth Network Communication System

A tenth network communication system of the present invention is arranged by the below-mentioned arrangement so as to solve the above-described second problem (corresponding to claim **12**).

That is, in the ninth network communication system, the security level control apparatus **10** owned by the client apparatus **30** may dynamically change the security level even during the communication in response to a request from the client apparatus **30**.

Then, the security level control apparatus **10** owned by the server apparatus **20** may dynamically change the security level even during the communication in response to a request from the server apparatus **20**.

In accordance with the tenth network communication system of the present invention, the following operations are carried out. That is, the security levels are dynamically variable even during the communication.

(Embodiment Modes)

Various embodiment modes of the present invention will now be described with reference to the drawings.

(System Arrangement of Embodiment Mode)

A system of an embodiment mode is arranged by employing a server apparatus **20** (also, referred to as a "server"), as shown in FIG. **6**, a network **40** connected to this server apparatus, and a client apparatus **30** (also, referred to as a "client") connected to this network **40**.

In this system, a communication is established between the server apparatus **20** and the client apparatus **30**. To prevent wiretapping, falsification, and impersonation in the communication, five stages of security levels can be set in accordance with importance of communication contents.

It should be noted that although only one server apparatus **20** is indicated in FIG. **6**, a plurality of server apparatuses may be employed. Similarly, although only one client apparatus **30** is shown in this drawing, a plurality of client apparatuses may be employed.

(Security Level)

When the communication starts, the server apparatus **20** and the client apparatus **30** notify independently set security levels to the counter party. Therefore, the server apparatus **20** and the client apparatus **30** communicate in accordance with security levels determined based upon the mutual security levels.

As previously described, in accordance with this embodiment, the security levels may be set in the five stages. These five-staged security levels are set as follows:

Security Level "1"—Neither encryption nor authentication is performed. It is a so-called "normal communication".

Security Level "2"—Only encryption is carried out.

Security level "3"—Both encryption and user authentication are performed.

Security level "4"—Both encryption and server authentication are carried out.

It should be noted that this security level "4" is a security level equivalent to the security level "3".

Security level "5"—Encryption, user authentication, and server authentication are carried out.

It should also be noted that when a plurality of client apparatuses **30** are provided, the server apparatus **20** may communicate in response to the security levels independently set for the respective client apparatuses **30**.

It should be also noted that when a plurality of server apparatuses **20** are provided, the client apparatus **30** may communicate in response to the security levels independently set for the respective server apparatuses 20.

Then, as the security level, other items may be set as follows. That is, no encryption is carried out at the security levels "2" to "5", and alternatively, the client is authenticated at the security levels "2" to "5".

Furthermore, the expression "user authentication" involves authentication with employment of a password, and authentication with employment of a public key certification. This embodiment mode describes the authentication with employment of the public key certification.

(Arrangement of Server Apparatus 20)

The server apparatus 20 is arranged by employing a communication control unit 21 connected to the network 40, a service processing unit 22 connected to this communication control unit 21, a security level control apparatus 10 connected to this service processing unit 22, and a storage unit 23 connected to the service processing unit 22.

The communication control unit 21 controls the communication established between the server apparatus 20 and the network 40.

To accept various service requests issued from the server apparatus 20, the service processing unit 22 transmits/receives the data among the security level control apparatus 10, the communication control unit 21, and the storage unit 23.

The storage unit 23 stores therein information concerning a user secret key (SKm: "m" being subscript), a user certification (CERTm: "m" being subscript), and a certification of an issuing station (CERTca: "ca" being subscript). As this storage unit 23, for instance, a RAM (Random Access Memory), a semiconductor memory device, a magnetic disk storage apparatus, a magnetic tape recording apparatus, an M/O (Magneto-Optical) disk apparatus, and an IC card are employed.

The security level control apparatus 10 is an apparatus for controlling security of actually performed communications based upon the security level notified from the client apparatus 30 and the security level owned by the server apparatus 20 when the communication is commenced. An arrangement of the security level control apparatus 10 will be explained subsequent to the description about the arrangement of the client apparatus 30.

(Arrangement of Client Apparatus 30)

The client apparatus 30 is arranged by employing a communication control unit 31 connected to the network 40, a service processing unit 32 connected to this communication control unit 31, a security level control apparatus 10 connected to this service processing unit 32, and a storage unit 33 connected to the service processing unit 32.

The communication control unit 31 controls the communication established between the server apparatus 20 and the network 40.

To accept various service requests issued from the client apparatus 30, the service processing unit 32 transmits/receives the data among the security level control apparatus 10, the communication control unit 31, and the storage unit 33.

The storage unit 33 stores therein information concerning a server Public key (PKs: "s" being subscript), a server certificate (CERTs: "s" being subscript), a server secret key (SKs: "s" being subscript), and a certificate of a Certification Authority (CERTca: "ca" being subscript). At this storage unit 33, for instance, a RAM (Random Access Memory), a semiconductor memory device, a magnetic disk storage apparatus, a magnetic tape recording apparatus, an M/O (Magneto-Optical) disk apparatus, and an IC card are employed.

The security level control apparatus 10 is such an apparatus for negotiating the security level notified from the server apparatus 20 with the security level owned by the client apparatus 30 when the communication is commenced.

(Arrangement of Security Level Control Apparatus 10)

Since the security level control apparatus 10 provided with the server apparatus 20 is arranged similar to the security level control apparatus 10 employed in the client apparatus 30, the arrangement thereof will now be described without giving any discrimination.

As shown in FIG. 7, the security level control apparatus 10 is constituted by employing a control unit 16, a security level recognizing unit 11, a security level converting table unit 12, a security level setting unit 14, a security level notifying unit 15, an encryption processing unit 17, and an authentication processing unit 18.

The control unit 16 is connected to either the service processing unit 22 (in case of server apparatus 20) or the service processing unit 32 (in case of client apparatus 30), and also connected to the security level recognizing unit 11, the security level converting table unit 12, the security level reading unit 13, the security level setting unit 14, the security level notifying unit 15, the encryption processing unit 17, and the authentication processing unit 18. Then, the control unit 16 controls data transmitting/receiving operations among these units.

The security level recognizing unit 11 recognizes the security level notified from the communication party.

The security level converting table unit 12 sets the index of the server to 1 through 5, and also the index of the client to 1 through 5 in such a case that the security levels used in the network are set to five stages, i.e., 1 through 5, which all the servers and all the clients can own in order that any of these servers and clients can use this converting table. Then, the security level concerning table unit 12 is arranged in such a manner that any one of the 25 patterns in total can be obtained based upon the security levels requested by the respective servers and clients which actually perform the communications.

In FIG. 8, there is shown the security level converting table unit 12 according to this embodiment. In the case of FIG. 8, assuming now that the security level of the client is "2" and the security level of the server is "4", the security level of the actual communication becomes "4". It should be noted in this drawing that a portion indicated as "X" implies that no communication can be performed at the security levels set by the server apparatus 20 and the client apparatus 30. In other words, this "X" portion corresponds to such a case that the security levels cannot be controlled.

As described above, the security level converting table unit 12 according to this embodiment is arranged as the following table, considering that the information provided by the server is important. That is, when the security level requested by the server is higher than the security level requested by the client, the security level requested by the server may have a priority.

However, the structure of the security level converting table unit 12 is not limited to the above-described embodiment. Alternatively, for example, the security level converting table unit 12 may be arranged by that only a security level which can be required by an own apparatus is set as a first index, and all of security levels which can be required by a counter party's apparatus are set as a second index. For instance, assuming now that in the above-described network, the own apparatus corresponds to the client which can require the security levels 1 through 3, and that the counter party's apparatus corresponds to the server which can require the security levels 1 through 5, any one of 15 patterns may be obtained, namely 15 patterns (in total)=indexes (3) of own apparatus x indexes (5) of the server.

While using the security level of the communication party recognized by the security level recognizing unit 11 and the security level for the security level control apparatus 10, as the index, the security level reading unit 13 reads a security level corresponding to this index from the security level converting table unit 12.

The security level setting unit 14 sets the security level read out from the security level reading unit 13 as the security level for the security level control apparatus 10.

The security level notifying unit 15 notifies the own security level to the communication party.

The encryption processing unit 17 encrypts a message to be outputted to the communication party, and conversely, decrypts the encrypted message entered from the communication party. It should be understood in this embodiment that the DES (Data Encryption Standard) system is utilized as the secret key cryptosystem, whereas the RAS (Rivest-Shamir-Aldeman) system is employed as the public key cryptosystem.

The authentication processing unit 18 performs server authentication (in case of client apparatus 30), and user authentication.

(Sequential Process Operation Between Client Apparatus 30 and Server Apparatus 20)

Referring now to FIG. 9 and FIG. 10, a description will be made of sequential process operation between the client apparatus 30 and the server apparatus 20 in the embodiment mode. It should be understood that all of the sequential process operations are not executed in this explanation, but only necessary process operations are executed every security level.

First, the client apparatus 30 notifies a communication request to the server apparatus 20 (step 901, this notification is expressed as "1"). In response this communication request, the server apparatus 20 notifies acceptance to the client apparatus 30 (step 902, this notification is indicated as "2").

After the acceptance is notified to the client apparatus 30, a communication preprocess operation is carried out between the client apparatus 30 and the server apparatus 20 (step 903, this preprocess operation is indicated by "3"). In this case, the communication preprocess operation implies information exchanges, for instance, information about terminal type, information about display system (how information is displayed by which line, which digit), information about sort of used character code, and IP address.

After the preprocess operation is complete, the client apparatus 30 notifies the security level set by the client apparatus 30 to the server apparatus 20 (step 904, this notification is indicated by "4"). Upon receipt of this notification, the server apparatus 20 recognizes the security level set by the client apparatus 30 (step 905, this recognition is indicated by "6").

Subsequently, the sever apparatus 20 notifies the security level set by the server apparatus 20 to the client apparatus 30 (step 906, this notification is indicated by "5"). Upon receipt of this notification, the client apparatus 30 recognizes the security level set by the server apparatus 20 (step 907, this recognition is denoted by "7").

In accordance with the security level set by the client apparatus 30 and the security level notified from the server apparatus 20, the client apparatus 30 selects the security level of the actually performed communication (step 908, this selection is expressed by "8").

In accordance with the security level set by the server apparatus 20 and the security level notified from the client apparatus 30, the server apparatus 20 selects the security level of the actually performed communication (step 909, this selection is expressed by "8").

It should be understood that the security levels selected at the step 908 and the step 909 are coincident with each other.

Thereafter, the client apparatus 30 notifies the user certification (CERTm) to the server apparatus 20 (step 910, this notification is expressed by "A").

The server apparatus 20 verifies the notified user certificate based on the certificate of the Certification Authority (CERTca) (step 911).

The server apparatus 20 notifies the public key (PKs) of the server, or the certification (CERTs) of the server to the client apparatus 30 (step 912, this notification is indicated by "B").

The client apparatus 30 verifies the notified certificate (CERTs) of the server based upon the certificate (CERTca) of the issuing Certification Authority step 913).

Also, the client apparatus 30 produces "DEK1" corresponding to seed for authentication (in this case, authentication of server) by way of random numbers (step 914).

Thereafter, the client apparatus 30 notifies to the server apparatus 20, PKs (DEK1) produced by encrypting "DEK1" based upon the public key (PKs) of the server (step 915, this notification is indicated by "C"). In other words, the client apparatus 30 corresponding to a "sender" encrypts a session key used to read the statement based on the public key (PKs) of the server apparatus 20 corresponding to a "receiver". Up to the present processing stage, since there is no session key for the client apparatus 30 and the server apparatus 20, the encryption is carried out by way of the public key cryptosystem (RSA).

The server apparatus 20 derives DEK1 by decoding the notified PKs (DEK1) by the secret key (SKs) of the server (step 916). In other words, the server apparatus 20 functioning as the receiver decodes the session key by using the secret key (SKs) of the server corresponding to the own secret key. Thereafter, the content sent from the client apparatus 30 is decoded by the decoded session key.

The server apparatus 20 produces SKs(DEK1) by performing DEK1 with employment of the server secret key (SKs) (step 917).

Thereafter, the server apparatus 20 notifies DEK1 (SKs (DEK1)) produced by encrypting SKs(DEK1) by DEK1 to the client apparatus 30 (step 918, this notification is indicated by "D"). In this case, the reason why SKs(DEK1) is encrypted by DEK1 is that an electronic signature is not wiretapped. The reason why such an electronic signature is made is to investigate that the sender (user) is authenticated and the content of the statement is not falsified. For example, a signer "A" makes up a digest of the statement by using a proper hash function, and then encrypts this digest by employment of a secret key for this signer "A". This may constitute a signature. A verifier "B" verifies the signature by employing the public key of the signer "A" to be returned to the original signature so as to check whether or not this result is equal to the digest of the original statement. If this result is not equal to the digest of the original statement, then it can be seen that the statement is falsified.

Now, the client apparatus 30 executes the following items 1) to 3) as a process "P" (step 919). 1). DEK1(SKs(DEK1)) is decoded to derive SKs(DEK1). 2). DEK1 is derived from the derived SKs(DEK1) by employing the public key (PKs) contained in the certificate of the server. 3). The derived DEK1 is compared with DEK1 produced at the step 914.

With this comparison, the server is authenticated. The reason why this authentication is perform is to confirm as to whether or not the public key opened as the server certificate is really the key for-the server apparatus 20. This confirmation is performed by employing the server certificate (CERTs) authenticated by a third party. Such a confirmation is also called as "third party authentication", or "electronic notary public". Simply speaking, a counter party makes a signature on a mail sent by an owner, and if this signature decrypted by employing the public key of the third party is identical to the signature sent by the owner, then the authentication can be established.

As a comparison result of the item 3) at the step 919, if the received signature is identical to the original signature, then the client apparatus 30 notifies "ACK" to the server apparatus 20, whereas if the received signature is not identical to the original signature, then the client apparatus 30 notifies "NACK" to the server apparatus 20.

Next, the server apparatus 20 produces DEK2 corresponding to a seed for authentication (in this case, authentication of user) by using random numbers (step 921).

Subsequently, the server apparatus 20 notifies to the client apparatus 30, DEK1(DEK2) produced by encrypting DEK2 by utilizing the secret key cryptosystem (step 922, this notification is indicated by "F"). In this case, the reason why the secret key cryptosystem is employed is such that the encryption key DEK1 is commonly used in the client apparatus 30 and the server apparatus 20, and when this encryption key DEK1 is utilized, the processing speed can be increased. In other words, if all of the encryption is done by Public key Cryptosystem.

Subsequently, the client apparatus 30 derives DEK2 by decoding the DEK1(DEK2) notified from the server apparatus 20 by way of the secret key (SKm) of the user (step 923).

Also, the client apparatus 30 produces SKm(DEK2) by making an electronic signature with respect to DEK2 by employing the secret key (SKm) of the user (step 924).

Thereafter, the client apparatus 30 notifies DEK2(SKm (DEK2)) produced by encrypting SKm(DEK2) by using DEK2 (step 925, this notification is expressed by "G").

Now, the server apparatus 20 executes the following items 1) to 3) as a process "Q" (step 926). 1). DEK2(SKm(DEK2)) is decoded to derive SKm(DEK2). 2). DEK2 is derived from the derived SKm(DEK2) by employing the user secret key (SKm). 3). The derived DEK2 is compared with DEK2 produced at the step 921. With this comparison, the user is authenticated.

As a comparison result of the item 3) at the step 926, if the decrypted signature is identical to the original signature, then the server apparatus 20 notifies "ACK" to the client apparatus 30, whereas if the received signature is not identical to the original signature, then the server apparatus 20 notifies "NACK" to the client apparatus 30 (step 927, this notification is repressed by "H").

Thereafter, a communication is carried out by employing the session key DEK2 between the client apparatus 30 and the server apparatus 20 (step 928, this communication is indicated by "9").

(Sequential Process Operations Executed In Respective Security Levels)

The sequential process operations executed in the respective security levels will now be explained with reference to FIG. 11. First, in the security level "1", the above-explained process operations (1), (2), (3), (4), (5), (6), (7), (8) and (9) are carried out in this order.

Next, in the security level "2", the above-described process operations (1), (2), (3), (4), (5), (6), (7), (8), (B), (C) and (F) are performed in this order.

Then, in the security level "3", the above-described process operations (1), (2), (3), (4), (5), (6), (7), (8), (A), (B), (C), (F), (G) and (H) are sequentially executed.

Then, in the security level "4", the above-described process operations (1), (2), (3), (4), (5), (6), (7), (8), (B), (C), (D), (E) and (F) are sequentially executed.

Next, in the security level "5", the above-described process operations (1), (2), (3), (4), (5), (6), (7), (8), (B), (C) and (D), (E), (F), (G) and (H) are performed in this order.

(First Process Operation)

Referring now to FIG. 12, the first process operation will be explained.

First, the client apparatus 30 (client) accesses a communication party (step 1201).

Next, the server apparatus 20 (server) accepts the communication by the client (step 1202).

At this stage, the communication preprocess operation is complete (step 1203).

Thereafter, the client requests the security level "3" (step 1204).

In response to this request, the security level control apparatus 10 of the server recognizes that the security level of the client is equal to "3" (step 1205).

Next, the server requests the security level "5" (step 1206).

In response to this request, the security level control apparatus 10 of the server recognizes that the security level of the client is equal to "5" (step 1207).

At this stage, the security level control apparatuses 10 of the server and the client select the security level "5" in accordance with the security level converting table unit 12 (step 1208).

Both the server and the client perform the encryption communication after executing the sequential process operations (A), (B), (C), (D), (E), (F), (G) and (H) of FIG. 9 and FIG. 10, and also exchange of the session keys (step 1209).

(Second Process Operation)

Referring now to FIG. 13, the second process operation will be explained.

First, the client apparatus 30 (client) accesses a communication party (step 1301).

Next, the server apparatus 20 (server) accepts the communication by the client (step 1302).

At this stage, the communication preprocess operation is complete (step 1303).

Thereafter, the client requests the security level "2" (step 1304).

In response to this request, the security level control apparatus 10 of the server recognizes that the security level of the client is equal to "2" (step 1305).

Next, the server requests the security level "2" (In 1306).

In response to this request, the security level control apparatus 10 of the client recognizes that the security level of the server is equal to "2" (step 1307).

At this stage, the security level control apparatuses 10 of the server and the client select the security level "2" in accordance with the security level converting table unit 12 (step 1308).

Both the server and the client perform the encryption communication after executing the sequential process operations (B), (C), and (F) of FIG. 9 and FIG. 10, and also exchange of the session keys (step 1309).

As previously described in detail, the communication level is not determined based upon only the communication level requested by the counter party's apparatus, but the actual communication level is determined based on the communication levels requested by both parties' apparatuses in this embodiment. As a consequence, the following effects can be achieved. That is, in the case that this embodiment is applied to the Internet, when a communication is established between a server and a user apparatus (called as a "host" in the Internet field), the security level converting table unit 12 is arranged in such a manner that the level requested by the server owns a priority so as to avoid the problems even under such a condition that although the server wants to encrypt the information in order to avoid that other apparatuses may refer to this information, the user requests to communicate the "plain text".

Furthermore, the situation between the server and the client are changed, above mentioned effects of this invention can be achieved.

What is claimed is:

1. A security level control apparatus for controlling a security level of a communication established between communication parties, comprising:

security level recognizing means for recognizing a security level notified from a communication party;

security level converting table means for storing therein a relationship between an index having two sets of security levels and a security level of an actual communication;

security level reading means for setting the security level of the communication party recognized by said security level recognizing means and a security level owned by said security level control apparatus as said index, and for reading a security level corresponding to said index from said security level converting table means; and

security level setting means for setting the security level read from said security level reading as the security level of said security level control apparatus.

2. A network communication system provided with a server apparatus and a client apparatus, which perform a communication whose security level is set, wherein

said client apparatus includes a security level control apparatus; and

said security level control apparatus includes:

security level recognizing means for recognizing a security level notified from a communication party;

security level setting means for setting the security level recognized by said security level recognizing means as a security level for said client apparatus; and

security level converting table means for storing therein a relationship between an index having two sets of security levels and a security level of an actual communication.

3. A network communication system as claimed in claim 2 wherein

said security level control apparatus provided with said client apparatus controls a security level with respect to the at least one server apparatus.

4. A network communication system as claimed in claim 2 wherein

said security level control apparatus owned by said client apparatus includes:

security level converting table means for storing therein a relationship between an index constituted by two sets of security levels and a security level of an actual communication; and

security level reading means for setting the security level of said at least one server apparatus recognized by said security level recognizing means and a security level of said client apparatus designated as said index, and for reading a security level corresponding to said index from said security level converting table means, and

said security level setting means sets the security level read out from said security level reading means as a security level for said client apparatus.

5. A network communication system provided with a server apparatus and at least one client apparatus, which perform a communication whose security level is set, wherein

said server apparatus includes a security level control apparatus; and

said security level control apparatus includes:

security level recognizing means for recognizing a security level notified from a communication party;

security level setting means for setting the security level recognized by said security level recognizing means as a security level for said server apparatus; and

security level converting table means for storing therein a relationship between an index having two sets of security levels and a security level of an actual communication.

6. A network communication system as claimed in claim 5 wherein

said security level control apparatus provided with said server apparatus controls a security level with respect to said at least one client apparatus.

7. A network communication system as claimed in claim 5 wherein

said security level control apparatus owned by said server apparatus includes:

security level converting table means for storing therein a relationship between an index constituted by two sets of security levels and a security level of an actual communication; and

security level reading means for setting the security level of said at least one client apparatus recognized by said security level recognizing means and a security level of said server apparatus designated as said index, and for reading a security level corresponding to said index from said security level converting table means, and

said security level setting means sets the security level read out from said security level reading means as a security level for said server apparatus.

8. A network communication system provided with at least one server apparatus and a client apparatus, which perform a communication whose security level is set, wherein

said client apparatus includes a security level control apparatus,

said security level control apparatus includes:

security level recognizing means for recognizing a security level notified from a communication party; and

security level setting means for setting the security level recognized by said security level recognizing means as a security level for said client apparatus,

said security level control apparatus owned by said client apparatus includes:

security level converting table means for storing therein a relationship between an index constituted by two sets of security levels and a security level of an actual communication; and

security level reading means for setting the security level of said at least one server apparatus recognized by said security level recognizing means and a security level of said client apparatus designated as said index, and for reading a security level corresponding to said index from said security level converting table means,

said security level setting means sets the security level read out from said security level reading means as a security level for said client apparatus, and

said security level control apparatus owned by said client apparatus dynamically changes the security level in response to a request of said client apparatus even during executions of the communication.

9. A network communication system provided with a server apparatus and at least one client apparatus, which perform a communication whose security level is set, wherein

said server apparatus includes a security level control apparatus,

said security level control apparatus includes:

security level recognizing means for recognizing a security level notified from a communication party; and

security level setting means for setting the security level recognized by said security level recognizing means as a security level for said server apparatus,

said security level control apparatus owned by said server apparatus includes:

security level converting table means for storing therein a relationship between an index constituted by two sets of security levels and a security level of an actual communication; and

security level reading means for setting the security level of said at least one client apparatus recognized by said security level recognizing means and a security level of said server apparatus designated as said index, and for reading a security level corresponding to said index from said security level converting table means,

said security level setting means sets the security level read out from said security level reading means as a security level for said server apparatus, and

said security level control apparatus owned by said server apparatus dynamically changes the security level in response to a request of said server apparatus even during executions of the communication.

10. A network communication system provided with a server apparatus and a client apparatus, which perform a communication whose security level is set, wherein:

said client apparatus and said server apparatus include security control apparatuses; and

each of said security level control apparatus includes:

security level recognizing means for recognizing a security level notified from a communication party;

security level converting table means for storing therein a relationship between an index constituted by two sets of security levels and a security level of an actual communication;

security level reading means for setting the security level of the communication party recognized by said security level recognizing means and a security level for said security level control apparatus as said index, and for reading a security level corresponding to said index from said security level converting table means; and

security level setting means for setting the security level read from said security level reading as the security level for said security level control apparatus.

11. A network communication system provided with at least one server apparatus and a client apparatus, which perform a communication whose security level is set, wherein

said client apparatus includes a security level control apparatus,

said security level control apparatus includes:

security level recognizing means for recognizing a security level notified from a communication party; and

security level setting means for setting the security level recognized by said security level recognizing means as a security level for said at least one server apparatus,

said security level control apparatus owned by said client apparatus includes:

security level converting table means for storing therein a relationship between an index constituted by two sets of security levels and a security level of an actual communication; and

security level reading means for setting the security level of said at least one server apparatus recognized by said security level recognizing means and a security level of said client apparatus designated as said index, and for reading a security level corresponding to said index from said security level converting table means,

said security level setting means sets the security level read out from said security level reading means as a security level for said client apparatus,

said security level control apparatus owned by said client apparatus dynamically changes the security level in response to a request of said client apparatus even during executions of the communication, and

said security level control apparatus owned by said server apparatus dynamically changes the security level in response to a request of said server apparatus even during executions of the communication.

12. A security level control method for controlling a security level of a communication established between communication parties, comprising the steps of:

recognizing a security level notified from a communication party;

storing a relationship between an index having two sets of security levels and a security level of an actual communication;

setting the security level of the communication party recognized by said recognizing step and a security level obtained by said security level control step designated as said index, and for obtaining a security level corresponding to said index from said storing step; and

setting the security level obtained from said setting step.

13. A network communication system having a server and client, comprising:

a security level table storing a relationship, based on a recognized security level of a communication party, between an index having two sets of security levels and a security level of an actual communication; and

a security level control apparatus controlling the security level of communications between the server and the client based on the security level table.

* * * * *

US005164938A

United States Patent [19]

Jurkevich et al.

[11] **Patent Number: 5,164,938**

[45] **Date of Patent: Nov. 17, 1992**

[54] **BANDWIDTH SEIZING IN INTEGRATED SERVICES NETWORKS**

[75] Inventors: **Mark Jurkevich**, Burtonsville, Md.; **Simon Bernstein**, Reston, Va.

[73] Assignee: **Sprint International Communications Corp.**, Reston, Va.

[21] Appl. No.: **676,539**

[22] Filed: **Mar. 28, 1991**

[51] **Int. Cl.^{5}** **H04J 3/16; H04J 3/00**

[52] **U.S. Cl.** **370/60**; 370/94.1; 370/94.2; 370/95.1; 370/60.1; 370/84

[58] **Field of Search** 370/94.1, 95.1, 60, 370/60.1, 94.2, 84

[56] **References Cited**

U.S. PATENT DOCUMENTS

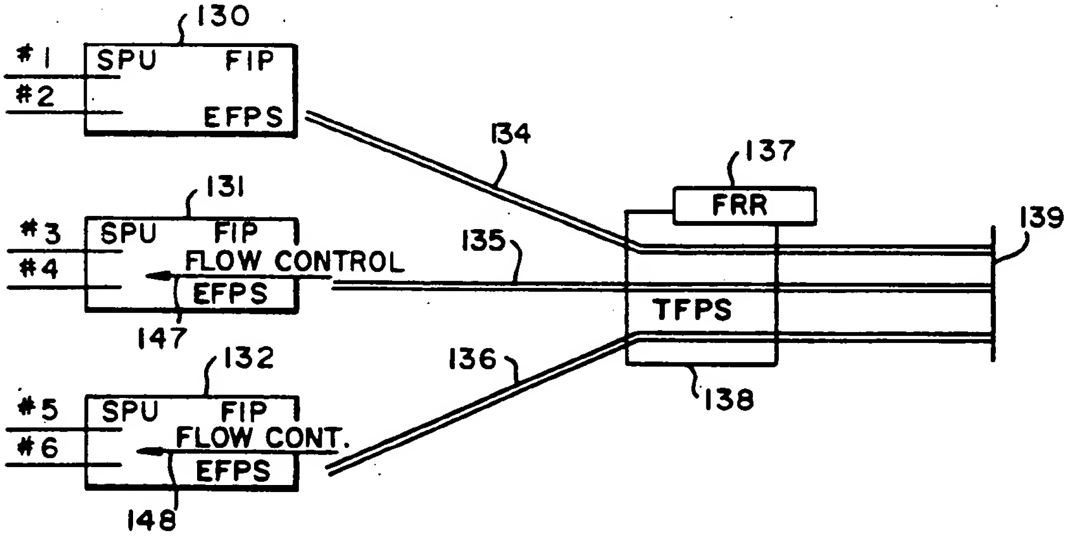
4,769,811	9/1988	Eckberg, Jr. et al.	370/60
4,774,706	9/1988	Adams	370/60
4,870,408	9/1989	Zdunek et al.	370/95.1
4,870,641	9/1989	Pattavina	370/94.1
4,893,305	1/1990	Fernandez et al.	370/84
4,942,569	7/1990	Maeno	370/60

Primary Examiner—Douglas W. Olms
Assistant Examiner—Russell W. Blum
Attorney, Agent, or Firm—Leitner, Greene & Christensen

[57] **ABSTRACT**

Method and system for transmitting information during call connections between a multiplicity of subscribers as components of traffic in an integrated services network (ISN), in which the information traffic consists of a multiplicity of media types according to the different subscribers including voice, video and data traffic component types. A plurality of traffic component types in the form of portions of information streams to be transmitted from subscribers at an entry point of the ISN during respective call connections are assembled into each of a sequence of composite frames of variable size for transmission through the ISN. The traffic component types assembled into each of the composite frames are limited to those destined for subscribers at the same exit point of the ISN. Each composite frame is configured with the traffic component types assigned to respective separate groups of adjacent channels of predetermined bandwidth with each group limited to channels transporting traffic components of the same type and each channel in a group dedicated to a particular subscriber of the respective traffic component type for the duration of its respective call connection. Bandwidth in the composite frames is selectively seized for reallocation among the various traffic component types during periods of traffic congestion.

15 Claims, 10 Drawing Sheets



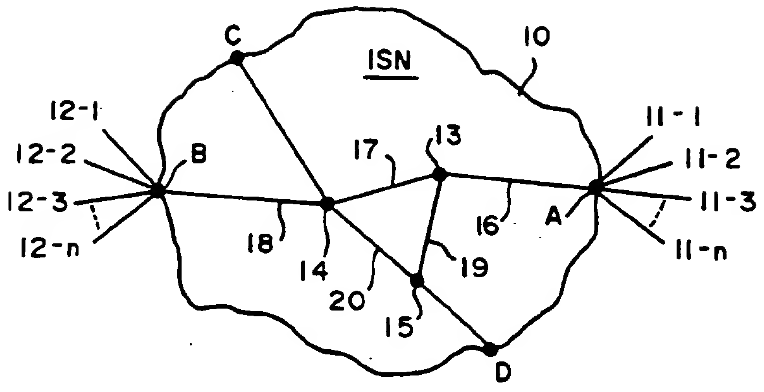
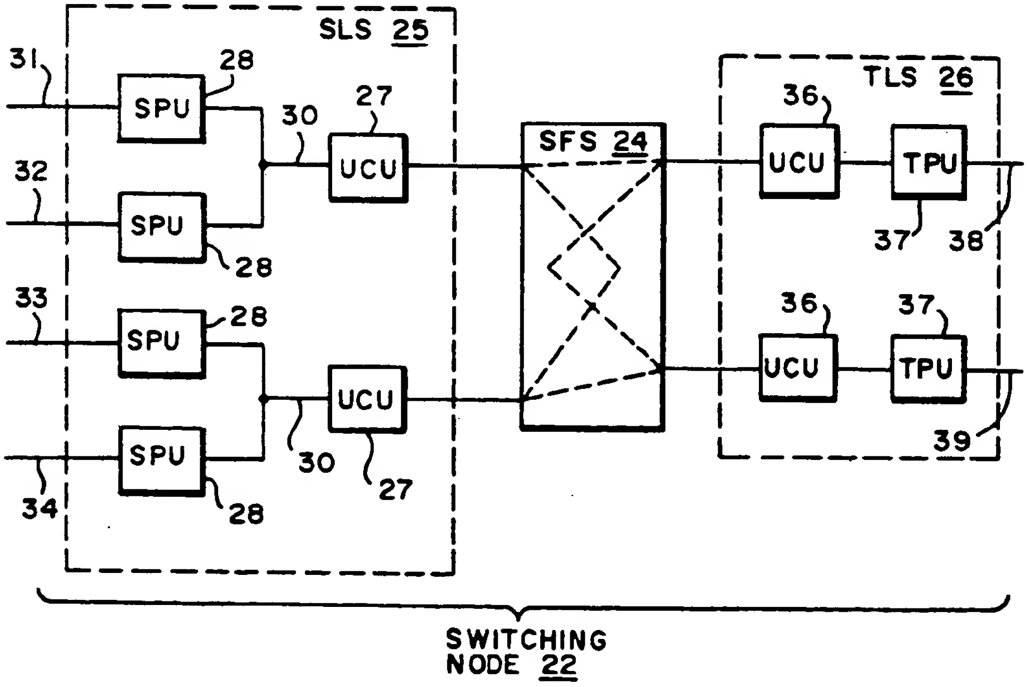

ISN
10
C
12-1
12-2
12-3
12-n
B
17
13
11-1
11-2
11-3
11-n
A
16
18
14
19
20
15
D

FIG. 1

FIG. 2


SLS 25
31
28
SPU
30
27
UCU
32
SPU
28
33
SPU
28
SPU
34
28
30
UCU
27
SFS 24
TLS 26
36
UCU
TPU
37
38
36
37
UCU
TPU
39
SWITCHING
NODE 22

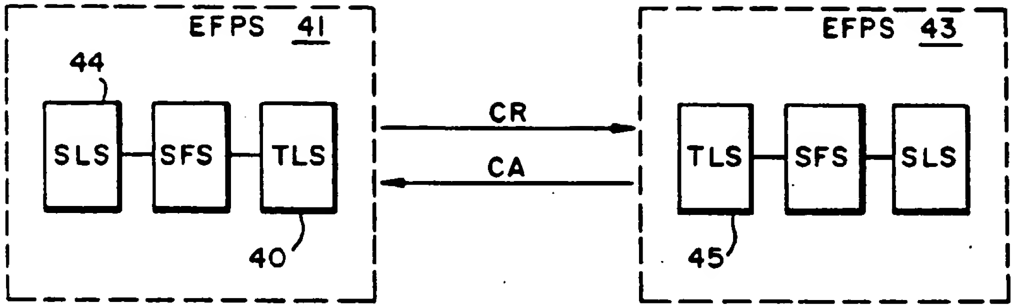

EFPS 41
44
SLS
SFS
TLS
40
CR
CA
EFPS 43
TLS
SFS
SLS
45

FIG. 3

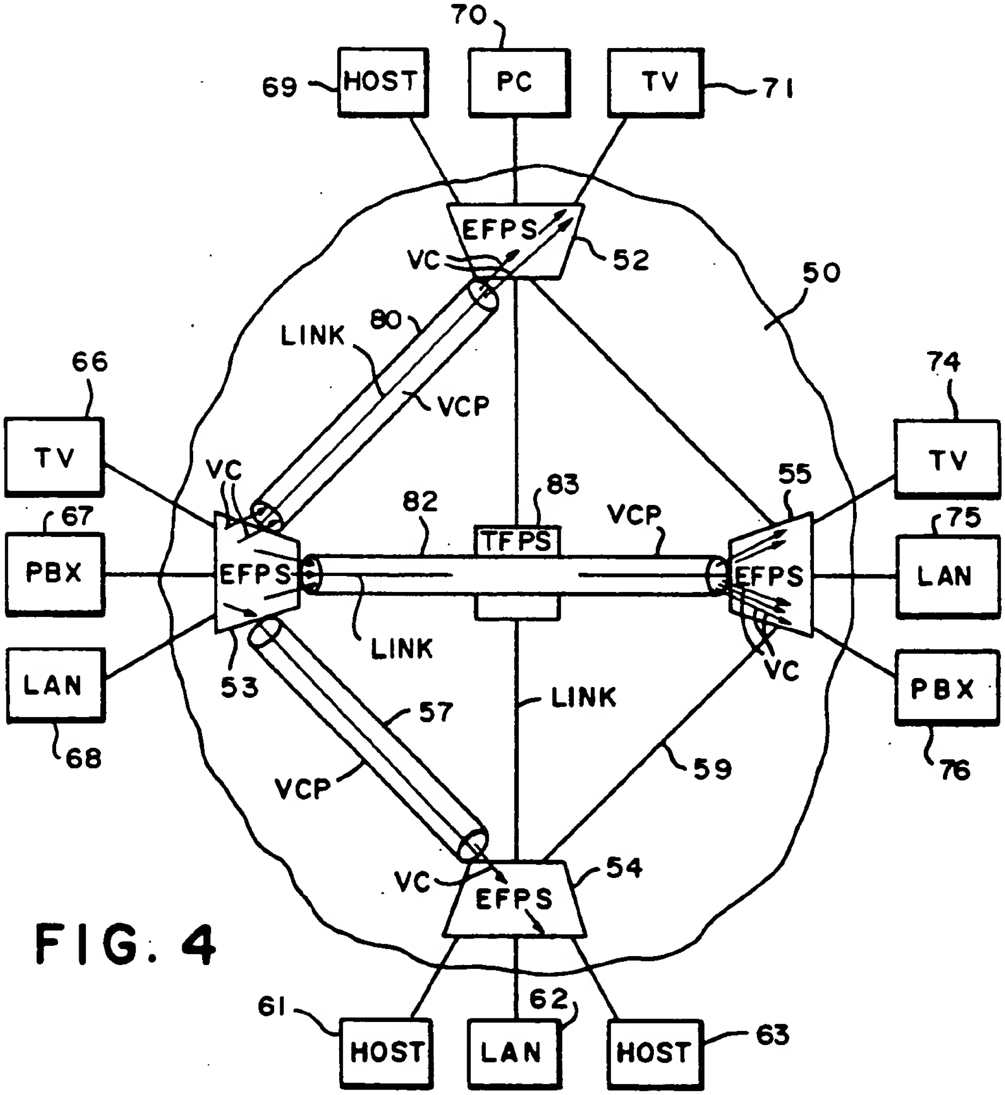

70
69
HOST
PC
TV
71
EFPS
VC
52
50
LINK
80
VCP
66
74
TV
TV
VC
82
83
VCP
55
67
TFPS
75
PBX
EFPS
EFPS
LAN
LINK
VC
LAN
53
LINK
PBX
57
68
59
76
VCP
VC
EFPS
54
61
62
HOST
LAN
HOST
63

FIG. 4

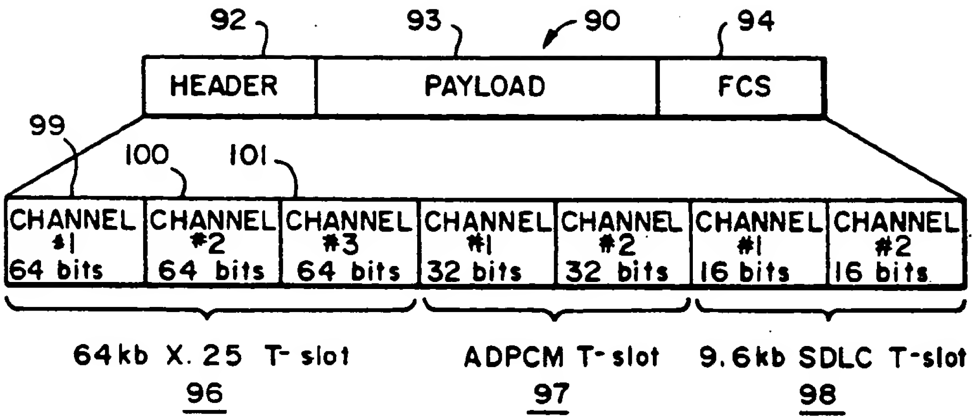


FIG. 5

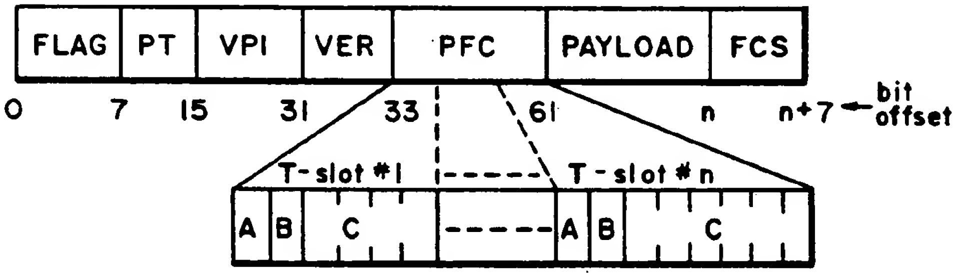


FIG. 6(a)

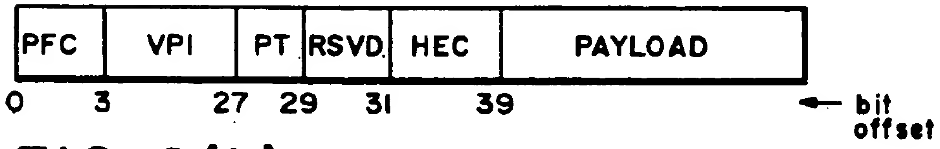


FIG. 6(b) (PRIOR ART)

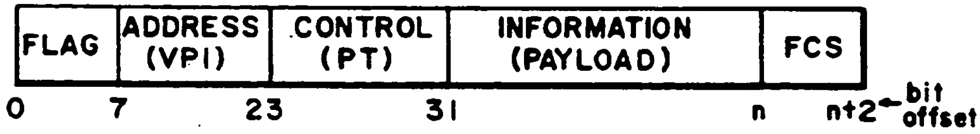


FIG. 6(c) (PRIOR ART)

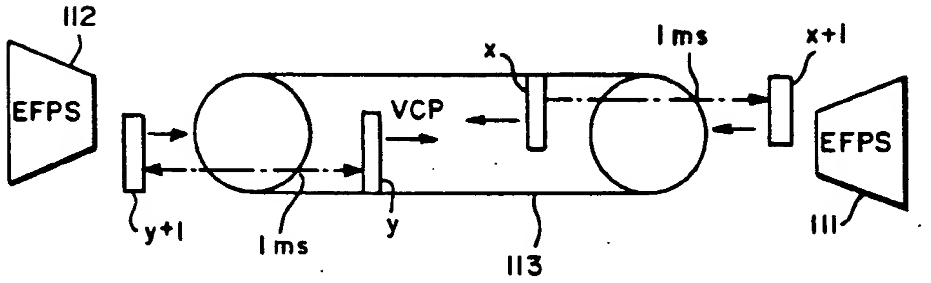


FIG. 7

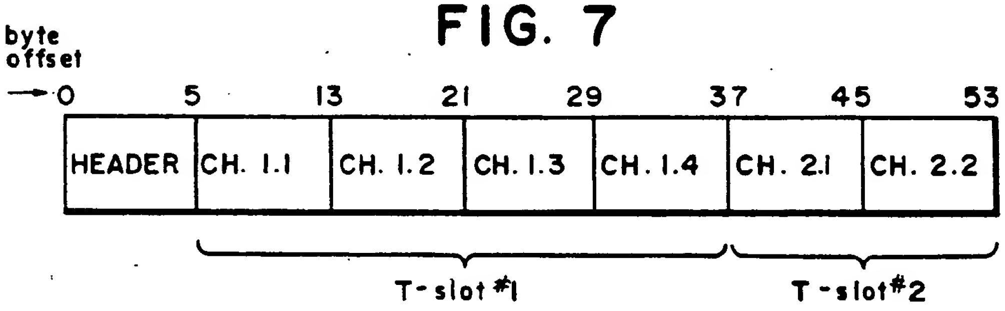


FIG. 8

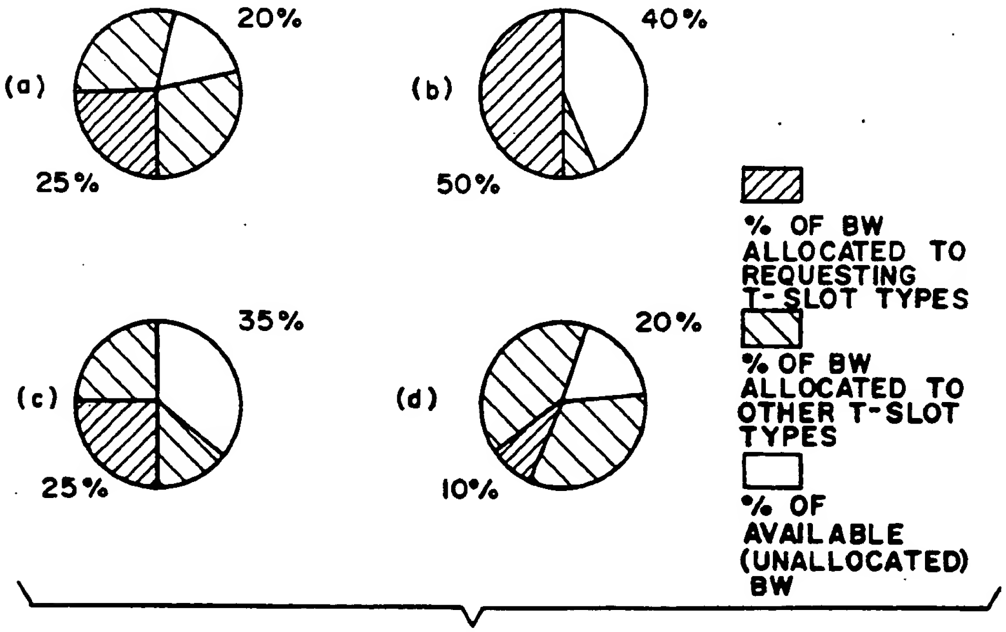


FIG. 9

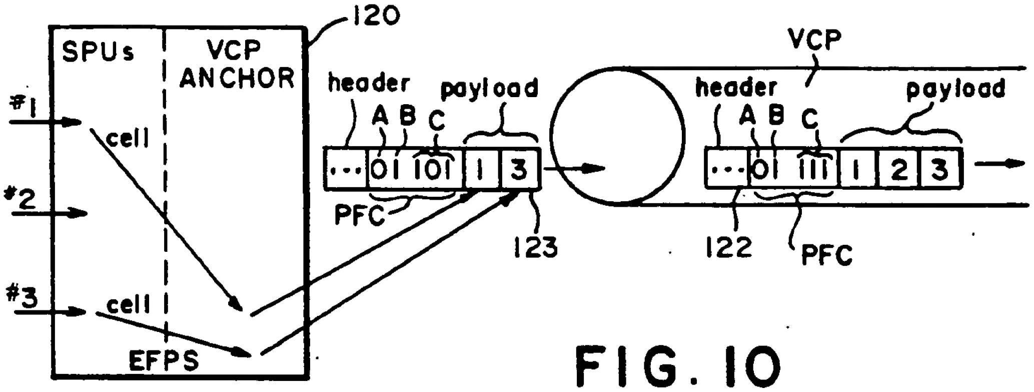


FIG. 10

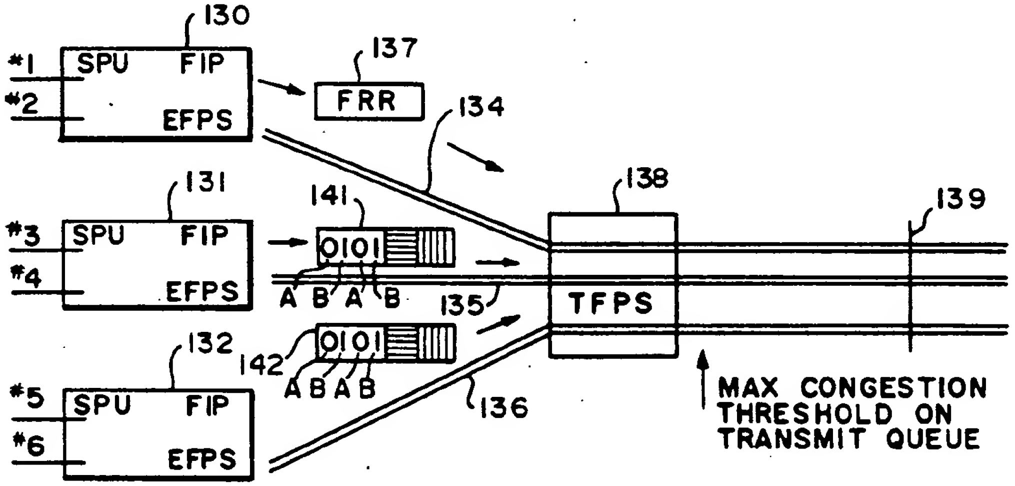


FIG. 11(a)

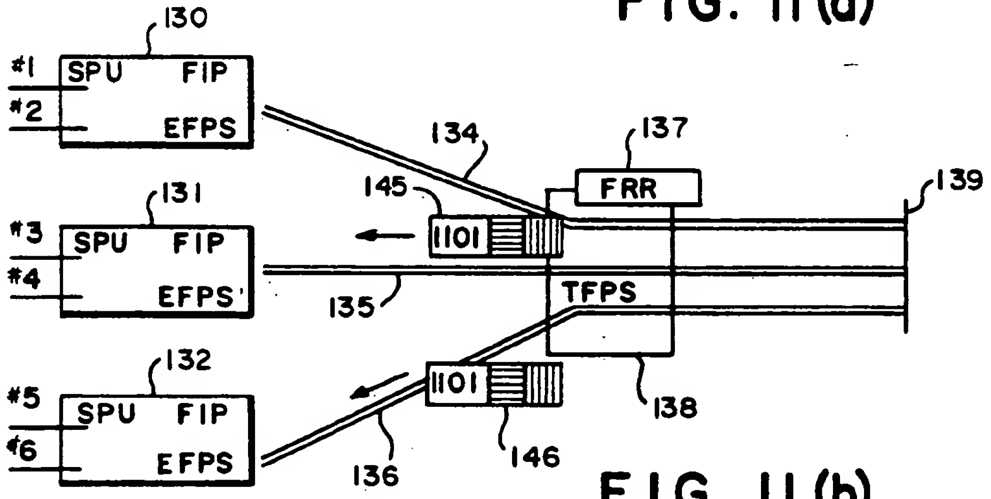


FIG. 11(b)

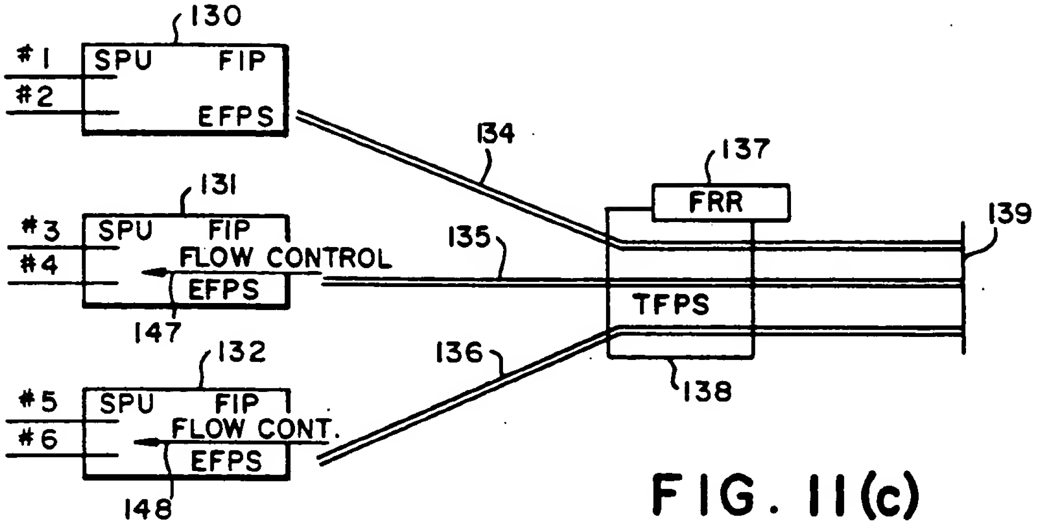


FIG. 11(c)

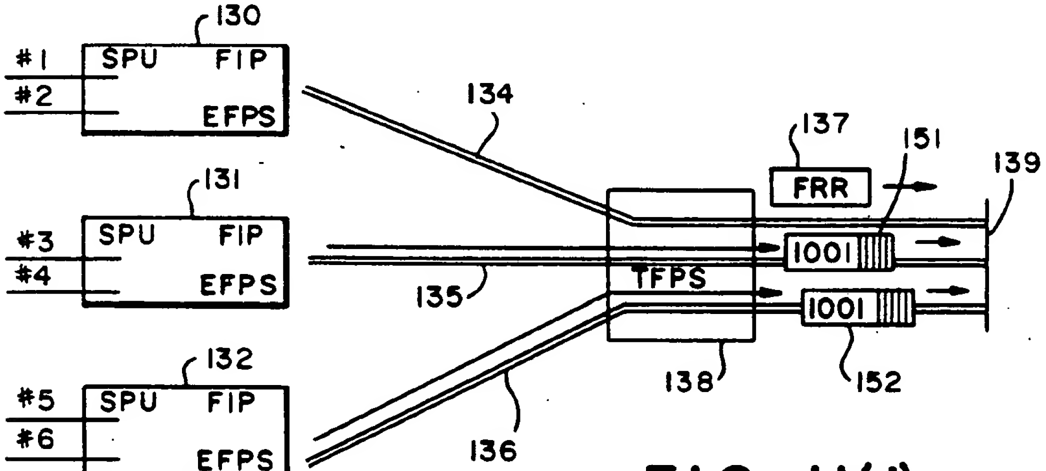


FIG. 11(d)

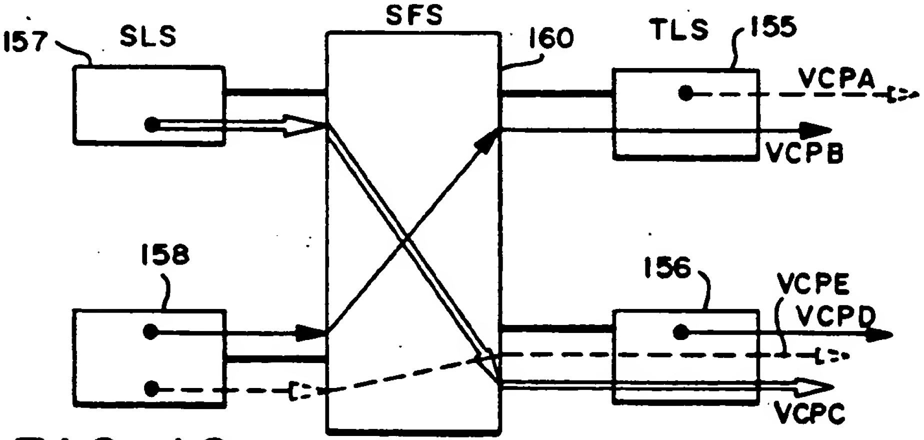


FIG. 12

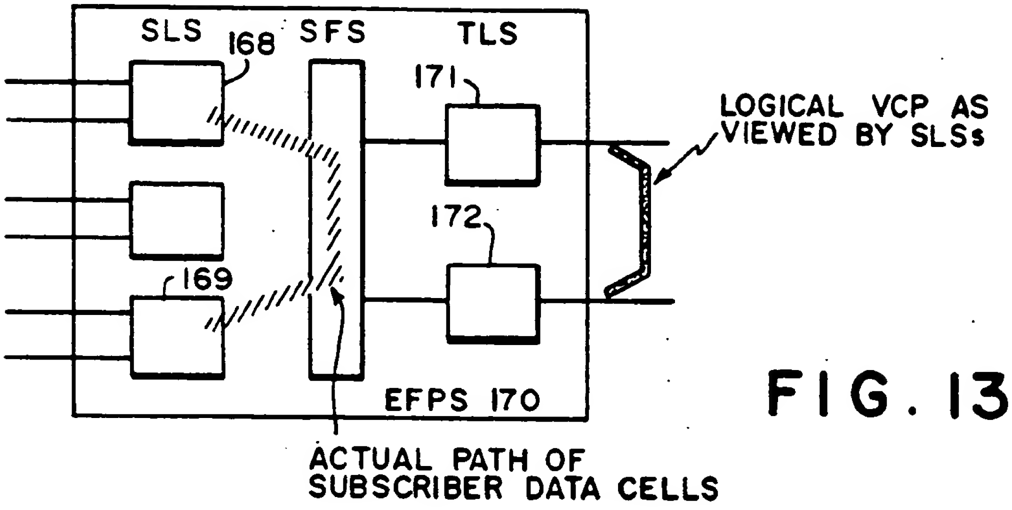

SLS
168
SFS
TLS
171
172
169
EFPS 170
LOGICAL VCP AS VIEWED BY SLSs
ACTUAL PATH OF SUBSCRIBER DATA CELLS

FIG. 13

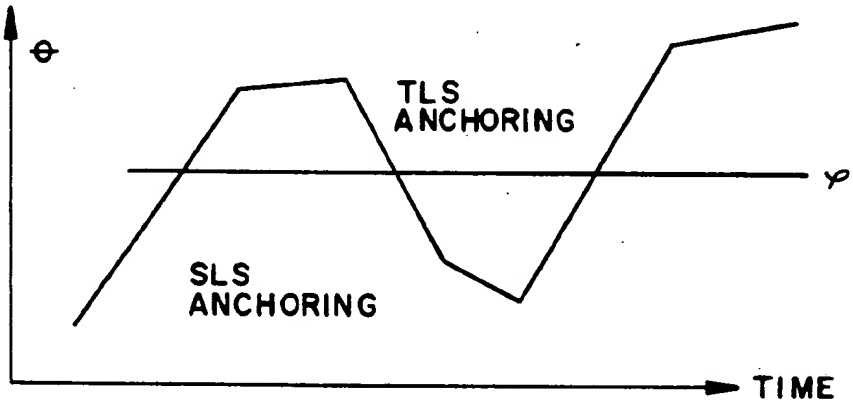


FIG. 14

#	LINK/T-SLOT PROFILE			A-BIT
	α	β	γ	
1	1	1	1	1
2	1	1	0	0
3	1	0	1	1
4	1	0	0	1
5	0	1	1	1
6	0	1	0	0
7	0	0	1	0
8	0	0	0	0

FIG. 18

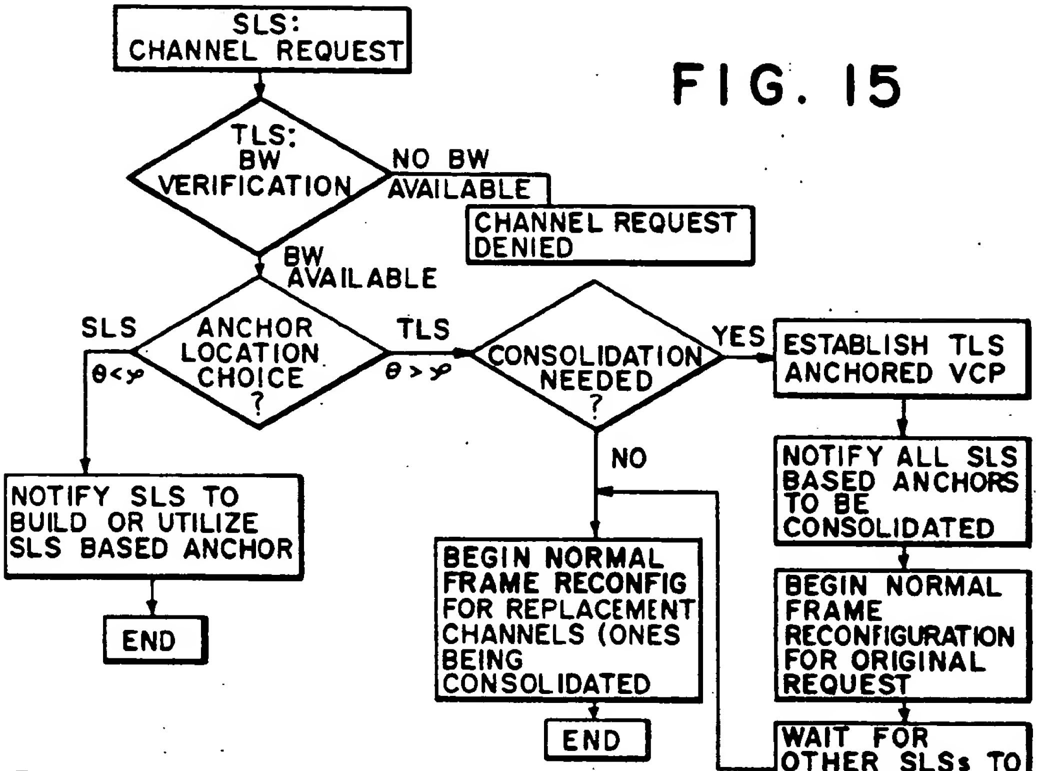

SLS:
CHANNEL REQUEST
TLS:
BW
VERIFICATION
NO BW
AVAILABLE
CHANNEL REQUEST
DENIED
BW
AVAILABLE
SLS
θ<φ
ANCHOR
LOCATION
CHOICE
?
TLS
θ>φ
CONSOLIDATION
NEEDED
?
YES
ESTABLISH TLS
ANCHORED VCP
NO
NOTIFY SLS TO
BUILD OR UTILIZE
SLS BASED ANCHOR
END
BEGIN NORMAL
FRAME RECONFIG
FOR REPLACEMENT
CHANNELS (ONES
BEING
CONSOLIDATED
END
NOTIFY ALL SLS
BASED ANCHORS
TO BE
CONSOLIDATED
BEGIN NORMAL
FRAME
RECONFIGURATION
FOR ORIGINAL
REQUEST
WAIT FOR
OTHER SLSs TO
REQUEST
REPLACEMENT
CHANNELS
(ONES BEING
CONSOLIDATED)

FIG. 15

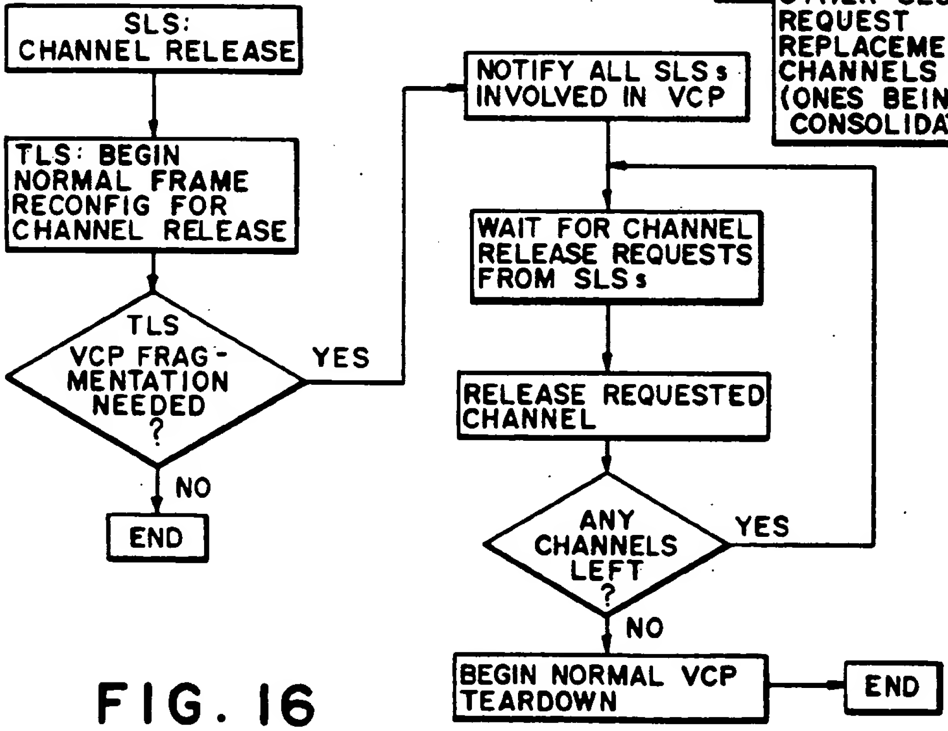

SLS:
CHANNEL RELEASE
TLS: BEGIN
NORMAL FRAME
RECONFIG FOR
CHANNEL RELEASE
TLS
VCP FRAG-
MENTATION
NEEDED
?
YES
NO
END
NOTIFY ALL SLSs
INVOLVED IN VCP
WAIT FOR CHANNEL
RELEASE REQUESTS
FROM SLSs
RELEASE REQUESTED
CHANNEL
ANY
CHANNELS
LEFT
?
YES
NO
BEGIN NORMAL VCP
TEARDOWN
END

FIG. 16

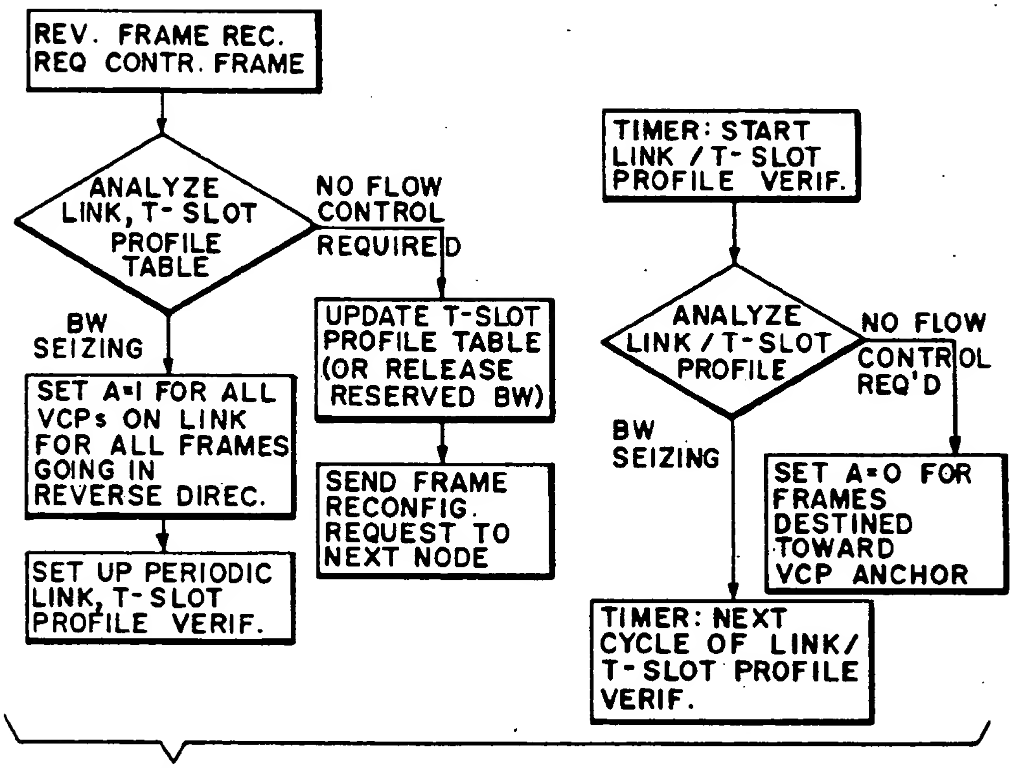

REV. FRAME REC.
REQ CONTR. FRAME
ANALYZE LINK, T-SLOT PROFILE TABLE
NO FLOW CONTROL REQUIRED
BW SEIZING
SET A=I FOR ALL VCPs ON LINK FOR ALL FRAMES GOING IN REVERSE DIREC.
SET UP PERIODIC LINK, T-SLOT PROFILE VERIF.
UPDATE T-SLOT PROFILE TABLE (OR RELEASE RESERVED BW)
SEND FRAME RECONFIG. REQUEST TO NEXT NODE
TIMER: START LINK / T-SLOT PROFILE VERIF.
ANALYZE LINK / T-SLOT PROFILE
NO FLOW CONTROL REQ'D
BW SEIZING
SET A=O FOR FRAMES DESTINED TOWARD VCP ANCHOR
TIMER: NEXT CYCLE OF LINK/ T-SLOT PROFILE VERIF.

FIG. 17

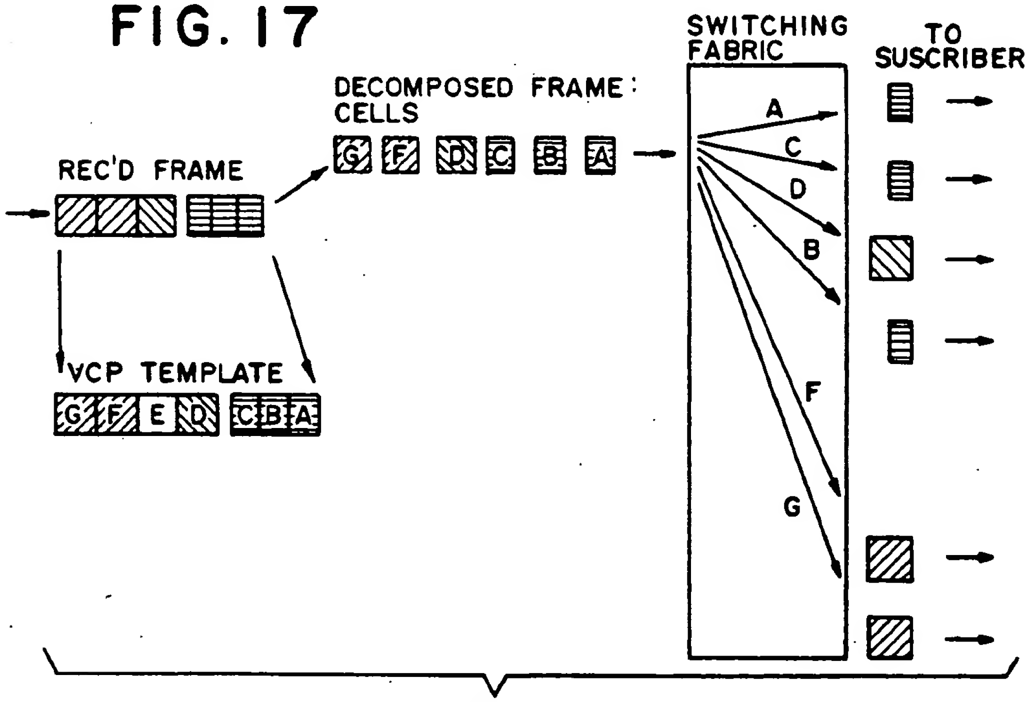

SWITCHING FABRIC
TO SUSCRIBER
DECOMPOSED FRAME: CELLS
G
F
D
C
B
A
REC'D FRAME
VCP TEMPLATE
G
F
E
D
C
B
A
A
C
D
B
F
G

FIG. 20

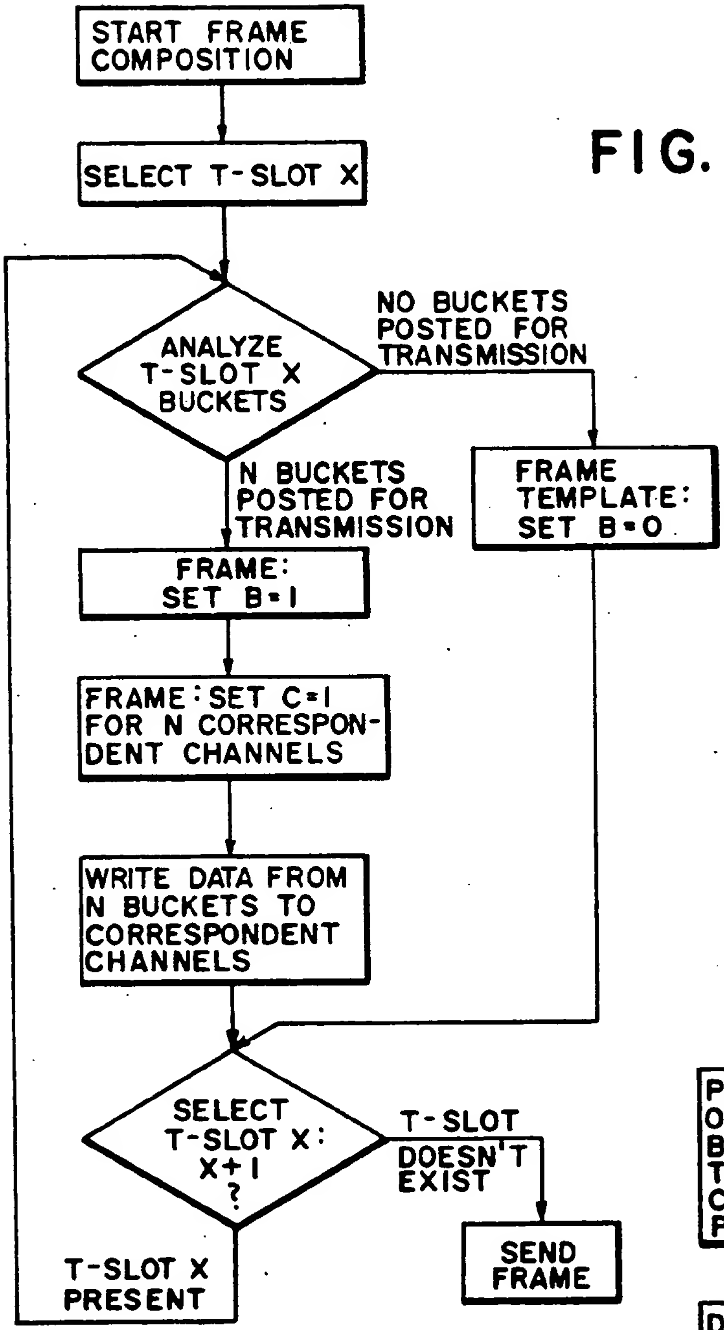


FIG. 19(a)

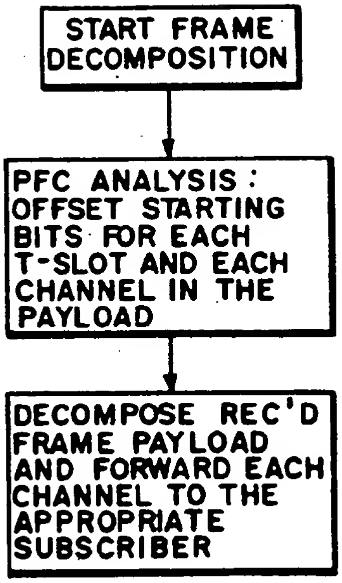

START FRAME DECOMPOSITION
PFC ANALYSIS: OFFSET STARTING BITS FOR EACH T-SLOT AND EACH CHANNEL IN THE PAYLOAD
DECOMPOSE REC'D FRAME PAYLOAD AND FORWARD EACH CHANNEL TO THE APPROPRIATE SUBSCRIBER

FIG. 19(b)

BANDWIDTH SEIZING IN INTEGRATED SERVICES NETWORKS

CROSS REFERENCE TO RELATED APPLICATIONS

This application is related to copending U.S. patent applications filed in the name of M. Jurkevich and S. Bernstein on even date herewith, and assigned to the same assignee as the instant application, as follows:

"Configurable Composite Data Frame", U.S. application Ser. No. 676,524;

"Frame Compression in Integrated Services Networks", U.S. application Ser. No. 676,535;

"Composite Frame Reconfiguration in Integrated Services Networks", U.S. application Ser. No. 676,537;

"Adaptive VCP Control in Integrated Services Networks", U.S. patent application Ser. No. 676,540;

"Prioritizing Attributes in Integrated Services Networks", U.S. application Ser. No. 676,515; and

"Fixed Interval Composite Framing in Integrated Services Networks", U.S. application Ser. No. 676,536.

BACKGROUND OF THE INVENTION

The present invention relates generally to packet switched digital telecommunication networks, and more particularly to improvements in fully integrated voice, data, and video (multimedia) communication services through the shared use of transmission and switching facilities in an integrated services network, including but not limited to networks such as those defined by the CCITT ISDN (Integrated Services Digital Network) and Broadband ISDN (B-ISDN) standards. The present invention provides for the coexistence and integration of 1.2 kilobits per second (kbps) to 2.045 megabits per second (mbps) applications with B-ISDN (>2.048 mbps) applications in a true multimedia network.

In recent years, the International Telegraph and Telephone Consultative Committee (CCITT), a telecommunications industry international standards-setting group, established Study Group 18 to undertake cooperative planning of B-ISDNs. A principal aspect of B-ISDN is the support it would offer to multimedia traffic applications, in which a multiplicity of traffic component types including voice, data, and video are to be communicated through the network. Each traffic component type exhibits significantly different characteristics or attributes from the others, and may have different characteristics among the members of its own type or class. For example, pure data traffic components may be interactive data, local file transfer data, facsimile data, and so forth, which have different burst sizes, or "burstiness". Such different attributes create differences in the requirements imposed on the network and local equipment for efficient and effective handling of the traffic component types in the communication between sources and destinations of the traffic. For instance, isolated loss of voice packets may be tolerated in telephone communications because the listener can comprehend the overall tenor of the conversation despite these slight gaps. Although quality suffers, the "human ear" is quite forgiving in these circumstances. Delays between different voice packets, i.e., a change in the sequence of the packets from source to destination, however, is unacceptable. In contrast, transmission of data such as X0.25 packets may not be adversely affected by delay among packets in transmission, but the

loss of individual packets can prevent restoration of an entire message.

In 1988, CCITT Study Group 18 approved recommendation I.121 which identified Asynchronous Transfer Mode (ATM) as the target solution for implementing B-ISDNs. ATM is an asynchronous time division multiplexing technique employing fast packet switching which communicates user information through the network in fixed length packets (called "cells" in the ATM jargon) of 53 bytes each. One mission of the Study Group and its Working Party 8 has been to standardize B-ISDN user network interfaces, including one at 155 mbps and another at 600 mbps. The present focus of the industry, however, is on fast packet (broadband) switching products at 1.54 to 45 mbps. For multimedia networks, the ATM scheme advanced by Study Group 18 uses fixed size cells each of which is assigned to a single user or traffic component type. Depending on user requirements at a given time, considerable bandwidth may be unused because partially empty channels are being transmitted.

In U.S. Pat. No. 4,980,886 titled "Communication System Utilizing Dynamically Slotted Information" (the "'886 Patent"), assigned to the same assignee as the present application, S. Bernstein discloses a multimedia system in which each packet or frame has the same payload size, with a fixed number of slots assigned to users, and in which the slot assignments may be changed periodically to improve communication performance. These are composite frames, packing several users/traffic component types into each frame, rather than only one user per frame.

The invention disclosed in the '886 Patent departs from prior burst switching technology by distributing user payloads among the available slots in a multimedia frame based on the specified bandwidth requirements of each user. The slots, which constitute portions of the available bandwidth for each frame, are not necessarily occupied by respective users from start to finish of a transmission. Instead, each user is guaranteed a certain minimum amount of bandwidth and all users contend for any unused bandwidth in each frame, according to their individual needs. The sending side packet switch allocates bandwidth on a frame-by-frame basis, so that users may be moved from one slot to another or to several slots in mid-transmission (i.e., on a "per burst" basis).

In the invention of the '886 Patent, unused bandwidth is not locked out; if a particular user has nothing to send or is not using its minimum guaranteed bandwidth (total slot or slots), the respective slot or portion thereof is allocated to a user having need for it. As the circumstances change, the allocations change. The receiving side packet switch monitors the slots in each incoming frame to keep track of the arriving information (data, voice, video, and so forth) and its sources, and to dispatch the information to its proper destination. Thus, the invention of the '886 Patent provides an entirely controllable bandwidth in which users are assigned priority rights to particular slots, but, depending on each user's particular need for bandwidth, bursts or blocks of information are temporarily allocated to unused slots or unused space in slots on a frame-by-frame basis.

SUMMARY OF THE INVENTION

The present invention also utilizes a composite frame approach for fast packet multimedia or integrated services networks, but instead of users contending for bandwidth in each frame as in the invention of the '886 Patent, bandwidth is conserved and efficiently utilized in a different way—namely, through techniques of frame compression and bandwidth seizing. The concepts of bandwidth contention within a frame as disclosed in the '886 Patent, and frame compression as disclosed in this application and its related applications, are based in part on the relatively recent concept of packet switching using fixed sizes. For example, older packet switching techniques such as X0.25 use variable size packets. The ATM scheme employs fixed size cells (with its disadvantages), but is of only recent vintage. The present invention utilizes variable size packets or frames having fixed size channels, and a scheme by which frames may be compressed to conserve bandwidth rather than employing techniques of contention for the available bandwidth.

The terminology "composite data frame" or "composite frame" as used herein refers to frames or packets which are composed of multimedia information components, that is, different traffic component types assembled into a single frame for transmission between subscribers through the network, and which may utilize techniques of frame compression and bandwidth seizing according to the invention. Within that terminology it will be understood that the term "data" is used in a broad sense, encompassing all traffic component types rather than being restricted to pure data only, although in other instances herein the terminology "data" will be used in the narrower sense.

It is a principal object of the present invention to provide an improved method for multimedia frame configuration and transmission in integrated services networks (ISNs), including those of the ISDN type.

It is another broad object of the present invention to provide improved techniques for configuring the payload and control information of a multimedia composite frame for communication between subscribers in an integrated services network.

According to an important aspect of the present invention, all of the various traffic component types in the data streams from multiple subscribers are assembled into composite frames configured for transmission to other subscribers through the integrated services network in such a way as to provide optimum network utilization with minimum cost, and at the same time to satisfy the individual performance requirements of each of the particular traffic component types. The various subscriber data streams are combined by traffic component type at the entry point to the network, if destined for the same exit point. At the exit point, the individual traffic component types are dispersed in separate directions according to their predetermined destinations.

Each traffic component type, whether voice, video, low speed data, high speed data or otherwise, possesses different characteristics or attributes, such as length of burst, ability to tolerate delay, and so forth. The network itself also has different characteristics or attributes, such as the inherent tendency to introduce transmission delay, which impacts on the attribute of each of the various traffic components' capacity to tolerate delay. Another inherent or intrinsic network attribute is the tendency to cause data loss depending on the nature of the traffic in the network. The extent of data loss that a traffic component can suffer and still allow the network to provide adequate service also varies from traffic component to traffic component. The phenomenon that different components of traffic in an integrated services network are affected differently by transmission characteristics of the network is, in and of itself, well known. Proposals in the prior art to solve this problem, however, have proved inadequate.

The present invention, in part, is effective to decouple the traffic component attributes and the network attributes and provide priorities for individual network attributes on a traffic component basis. The principles employed, in which all network attributes are controllable entities on a per-traffic component basis, are to be contrasted with specialized network approaches employed in present day telecommunications systems, in which a single priority level scheme applies for all network attributes. The latter are truly effective where there is only one traffic component and only one or relatively few network attributes which apply to that component, such as in an X0.25 data network or a pure voice network. The present invention includes assigning of priorities so that, for example, voice traffic may be allowed to suffer data loss but no delays, while data packets such as X0.25 are permitted to suffer delay but no data loss. Such conflicting requirements are resolved in one aspect by assigning traffic component types to separate frames according to their respective sensitivities and tolerances, while satisfying the need for rapid transmission and increased throughput performance in the network.

It is therefore another object of the present invention to provide systems and methods in an integrated services network by which the transmission and throughput performance of various traffic component types is enhanced by prioritizing them on the basis of their respective attributes in the environment of the ISN, so that priority of transmission can be given to those composite data frames containing the traffic component types assigned the higher priorities during periods of traffic congestion or when traffic flow otherwise requires control.

According to a feature of the present invention, the multimedia communication method and system utilizes a composite data frame configured with a multi-slotted payload, each slot being a channel which is allocated to a subscriber having requirements for transmission of a particular type of traffic component. The payload of the composite frame is divided into multiple channels and the channels are grouped according to traffic component type, with each grouping of plural channels in the frame referred to herein a traffic component slot, or simply, T-slot. The frames are composed with a particular configuration of channel assignments and inclusions on a per call connection basis, dedicated for the duration of the call connection, and may be reconfigured on request by subscriber according to established priorities or based on traffic conditions such as link congestion on the network.

Present day schemes provide static allocation of channels, and contention for channels by active connections. In contrast, the present invention allocates channels dynamically upon request at connection activation time (and deallocates on call termination); and there is no contention for channels—rather, the channels are dedicated to one connection for the entire duration of that connection. The multimedia information (voice, data, video and/or other traffic component type) to be transmitted from multiple subscribers located at a network entry point is assembled from the subscriber data streams into fixed size packets for consolidation in the same size channels allocated to the subscribers in the payload of a composite frame, provided that the various traffic components are all destined for the same network exit point. That is, assignment of the various subscriber data streams (of like or varying T-slot types) to the payload of a composite frame for transmission through the network is limited to those traffic components which share the same source node and same destination node in the network.

Hence, another object of the invention is to provide a composite data frame of variable size which is configured as a vehicle to convey through the network data streams emanating from subscribers at a source endpoint node of the network, in the form of a plurality of traffic component types, in channels grouped and of fixed size according to traffic component type, provided that the traffic components assembled within any given composite frame are destined for the same endpoint node.

According to another feature of the invention, the composite frames are assembled by fixed interval framing and transmitted through the network by synchronous frame launching. To that end, each packet is shipped at a predefined fixed interval of time relative to the timing of shipment of the immediately preceding packet, without regard to whether or not each channel in the packet is completely filled at that point in time. The synchronous frame launching is used to build composite frames with fixed channel sizes, which permits elimination of overhead control information including specification of channel size, amount of information to be received, and maximum amount of information to be transmitted on the connection, typically associated with other existing composite frame schemes. This reduces the amount of bandwidth required for transmission of the frames.

Another object of the invention, therefore, is to provide a fast packet switched integrated services network in which composite frames are assembled and launched onto the network at fixed intervals of time, in which the fixed interval is consistent throughout the network.

Decomposition information is transmitted to the exit point for the composite frames in the network by specifying the number of channels being allocated and the traffic component type for each, in a separate control frame carried outside the composite data frames. The control frame is built by the local endpoint node and sent to the remote endpoint node, when a network subscriber requests a connection or termination of a connection. Each control frame is built to contain only the delta change from the prior frame format to the current frame format, identifying the channels being added or released in the composite frame to the network remote endpoint. When a channel or channels are added, the control frame must specify the traffic component type of each such channel.

According to an important aspect of the invention, if a subscriber is not fully active, in the sense that the information stream generated by that subscriber to be transmitted to the remote endpoint within the composite data frame being assembled at the local endpoint is inadequate to fill the channel allocated to that subscriber, that channel is eliminated from the frame. In this way, any unused bandwidth is compressed out of the composite frame payload before the frame is launched into the ISN.

A further object of the invention, then, is to provide bandwidth conservation in an integrated services network in which information is conveyed in the form of composite data frames containing a plurality of traffic component types, by a technique of compressing out of each frame any unused bandwidth.

Frame compression is one of three interrelated aspects of the invention which, however, may be employed independently in ISN FPS networks. The other two of this triumvirate are reconfiguration of the composite frames, and bandwidth seizing. As has been observed herein, the composite data frame is configured with the traffic component types assigned to respective separate groups of adjacent channels for each traffic component type, so that each group is limited to channels transporting traffic components of the same type, with each channel in a group assigned entirely to a selected subscriber associated with the traffic component type for that group. According to the invention, a composite frame is reconfigured to modify the channel assignments when necessary to accommodate priorities for traffic flow among the subscribers on a network path (virtual circuit path) between entry and exit points (the two endpoint nodes or fast packet switches of the virtual circuit path) of the ISN. Bandwidth seizing is implemented when, because of priority assignments among the various traffic component types relating to concepts of guaranteed bandwidth, and traffic congestion on the network or more specifically on links or trunks of the virtual circuit path of interett, bandwidth allocation is taken at least in part from one or more traffic component types and redistributed to another or other traffic component types.

Traffic flow control is initiated at a node along the network path of interest when a link on the path associated with that node exceeds a predetermined link utilization threshold level indicative of traffic congestion. Such flow control may be undertaken either when a request for additional bandwidth (i.e., the making available of a channel) is made by any traffic component type (or more specifically, a subscriber of that traffic component type) which is below its minimum guaranteed bandwidth, or when an unusually large number of subscribers at an endpoint node are simultaneously seeking to transmit information for assembly into composite data frames. The flow control affects those traffic component types which are exceeding their minimum guaranteed bandwidth, starting with those of lowest priority. For each composite data frame in the receive queue on the congested link of the affected transit node along the network path the node modifies a field in the header of the composite data frame to indicate that flow control is being exercised.

A reconfiguration request control frame is issued at the endpoint node of the subscriber needing additional bandwidth and meeting the necessary predetermined criteria. This request for additional bandwidth for the justified traffic component type will ultimately result in the seizure of bandwidth from any traffic component type which is exceeding its respective minimum guaranteed bandwidth in the composite data frames. At the endpoint node launching the composite frames to which the request applies, frame compression is implemented to unlock bandwidth by seizing it from the traffic component type(s) targeted by the reconfiguration request control packet. A less frequent posting of cells comprising portions of the information streams from the affected subscribers, for assembly into the composite frames, results in frame compression by eliminating some or all channels of the traffic component types associated with the excessive bandwidth usage in at least some of the composite data frames. The freed bandwidth is thereby reallocated or redistributed and the reconfiguration request control frame is dispatched to the next transit node along the network path when the traffic profile indicates that the associated link is no longer congested. The reconfiguration request control frame is a packet analogous to a call setup packet, and is transported along the same virtual circuit path as the composite data frames, but acts as a control element to change the format of the composite frames so long as the request is not blocked (rejected) by a node along the path. The existing format of the composite frames is contained in a template stored at each of the nodes along the path, and another stored template indicates the amount of change of bandwidth which is permitted for a particular traffic component type.

Therefore, yet another object of the invention is to provide a method and system for selectively reconfiguring composite data frames in an integrated services network as necessary for optimum bandwidth utilization, traffic flow and throughput performance.

Still another object is to provide a scheme for selectively seizing bandwidth from one or more traffic component types and redistributing the seized bandwidth to one or more other traffic component types having a greater priority for the bandwidth in an integrated services network.

According to still another aspect and feature of the invention, logical connections are established between subscribers at endpoint nodes of the ISN at the time of call setup, in the form of virtual circuits (VCs), and between pairs of endpoint nodes to accommodate a multiplicity of VCs, in the form of virtual circuit paths (VCPs), and the establishment, location and relocation of VCP anchors at endpoint nodes within the ISN are adaptively controlled according to the needs of the network and its subscribers. Each endpoint node, or more precisely the point of multiplexing within the node, may anchor more than one VCP. Each VCP not only constitutes a logical connection between a pair of endpoint nodes, but has a one-to-one coupling with the composite data frame transported on it.

Information concerning each VCP anchored at a particular endpoint node (a fast packet switch) is stored at that node. In some instances a VCP is anchored at the trunk side of the switch fabric, and in other instances a VCP is anchored at the subscriber side of the switch fabric. The decision on where to anchor the VCP in these instances is based on the traffic patterns between the source and destination endpoint nodes, and includes such factors as whether the VCs to be multiplexed terminate on the VCP anchor ŋode, whether all trunk line subsystems (TLSs) and subscriber line subsystems (SLSs) at the endpoint node have the capability of anchoring a VCP, and whether the subscriber data stream will pass through the switch fabric not more than once (except in the case of local switching).

The choices of whether to have multiple parallel VCPs between endpoint nodes and of where to locate the VCP anchor(s) within a particular endpoint node, are determined by the opportunity to multiplex VCs onto the VCP. Periodic reevaluation is performed within the ISN for optimal VCP anchor locations and VC loading (i.e., number of VCs multiplexed). As network traffic conditions change over time, the invention implements adaptive relocation of the VCP anchor to the optimal location for those conditions. Each endpoint node is made capable of rerouting VCPs, relocating VCP anchors, consolidating VCP anchors, and even subdividing a VCP. As the VC load increases between a pair of endpoint nodes, multiple SLS-anchored VCPs are consolidated into a single TLS-anchored VCP which uses the network-wide frame launch period. A TLS-anchored VCP may be converted to an SLS-anchored VCP when the VCP traffic load drops to a level in which the payload/header ratio of the composite data frames is unacceptably small. An existing VCP may be rerouted/reconnected if the existing route is not optimal for the network topology or traffic conditions.

According to this aspect of the invention, anchor relocation is triggered by either (1) relocation on demand, or (2) periodic relocation. In relocation on demand, anchor location is reevaluated during each VC call request from a subscriber. In periodic relocation, the relocation occurs at a fixed time or time interval. Periodic relocation is somewhat less likely to result in thrashing between anchor locations, than relocation on demand.

Accordingly, it is another object of the invention to provide methods and systems for adaptive control of VCPs in an integrated services network designed to transmit a multiplicity of traffic component types between endpoint nodes of the network within configurable composite data frames via VCPs established as logical connections between pairs of the endpoint nodes.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and still further objects, features, aspects and attendant advantages of the present invention will become apparent from a consideration of the following detailed description of a presently preferred method and embodiment of the invention, taken in conjunction with the accompanying drawings, in which:

FIG. 1 is a simplified diagram useful for explaining some of the basic concepts of the integrated services network environment in which systems and methods of the present invention may be used;

FIG. 2 is a simplified block diagram of the basic structure of a packet switch useful for implementing certain concepts the invention;

FIG. 3 is a block diagram of a pair of endpoint fast packet switches establishing a call connection, useful to explain source and destination designations on a VC or VCP;

FIG. 4 is a block diagram illustrating the relationship of VCs to VCPs;

FIG. 5 is a representation of an exemplary composite data frame according to the preferred embodiment and method of the invention, with a fixed payload size and composition accommodating a plurality of traffic component types;

FIG. 6 is a simplified comparison of three different packet types, the composite data frame according to the preferred embodiment and method of the present invention being shown in part (a), and the ATM cell and LAPD frame of the prior art being shown in parts (b) and (c), respectively;

FIG. 7 is a simplified block diagrammatic representation of a VCP with synchronous frame launching according to the invention;

FIG. 8 is a representation of a composite data frame which provides an illustrative example of payload size for a plurality of highly active subscribers;

FIG. 9 is a set of exemplary charts illustrating the disposition of bandwidth allocation requests (FRRs) under various traffic conditions, i.e., BW grant/reject scenarios;

FIG. 10 is a simplified diagram of a VCP anchor EFPS illustrating the launching of composite data frames utilizing the preferred frame compression method of the invention;

FIGS. 11(*a*)–(*d*) are a sequence of frame processing diagrams illustrative of the initiation of flow control through bandwidth seizing according to the invention;

FIG. 12 is a block diagram illustrating the technique for anchoring a VCP in an EFPS;

FIG. 13 is a block diagram useful for explaining a local switching example in VCP anchoring;

FIG. 14 is a graph illustrating a hypothetical case of the VCP anchoring process in real time;

FIGS. 15 and 16 are flow charts indicative of the processing required for adaptive anchoring of VCPs with relocation on request for a channel and release of a channel, respectively;

FIG. 17 is a flow chart illustrating the A bit set-up procedure for bandwidth seizing;

FIG. 18 is a table indicating an exemplary link/T-slot profile for A bit set-up conditions in conjunction with bandwidth seizing;

FIGS. 19(*a*) and (*b*) are flow charts illustrating the B and C bits set-up procedure for frame composition at the source node, and the PFC field and payload analysis for frame decomposition at the destination node; and

FIG. 20 is a simplified block diagram illustrating the retrieval and delivery of data from the received composite data frames by the destination node.

DESCRIPTION OF PRESENTLY PREFERRED EMBODIMENT AND METHOD

Referring to FIG. 1, a fast packet switch (FPS) network serving as an integrated services network (ISN) 10 of a type in which the present invention is employed transports multimedia information in data frames or packets, each possibly containing a plurality of traffic component types. The frames are transported at fast packet speeds between a pair of subscribers at endpoints of the network, such as endpoints A and B. Network 10 typically has a multiplicity of endpoints A, B, C, D, etc., each serving a plurality of subscribers, such as 11-1, 11-2, . . . , 11-n at endpoint A and 12-1, 12-2, . . . , 12-n at endpoint B. The actual number of subscribers served at the various endpoints of the network may differ from endpoint to endpoint.

According to an aspect of the invention, an endpoint fast packet node or switch (EFPS) is located at each endpoint, and a transit fast packet switch (TFPS, or sometimes referred to herein simply as a transit switch) is located at each of a multiplicity of intermediates nodes such as 13, 14 and 15, of network 10. Each transit switch accommodates a plurality of transmission links or trunks within network 10. Thus, a packet launched from endpoint A to endpoint B, for example, may travel through trunks 16, 17 and 18 across transit switches 13 and 14, or, depending upon the traffic conditions, through trunks 16, 19, 20 and 18 across transit switches 13, 15 and 14. Each EFPS and TFPS of the network is a packet switch in the form of a communication processor, but the EFPS and TFPS differ from one another in implementation or the algorithms they implement, as will be explained presently.

A logical connection established between two subscribers of the integrated services network through ordinary call set-up procedures is referred to herein as a virtual circuit (VC). For example, a VC is established between subscribers 11-1 and 12-3 for a call (communication session) between the two, and remains in place for the duration of that call. To reduce individual call processing, a plurality of VCs which share a single source-destination EFPS pair may be routed (actually, multiplexed) by defining an end-to-end network path for them. Each such network path constitutes a single physical link referred to herein as a virtual circuit path (VCP). Thus, each VCP defines a logical connection between a particular pair of EFPSs such as the EFPS at endpoint A and the EFPS at endpoint B, or more specifically, between the points of VC multiplexing within the two EFPSs, in contrast to the logical connection between two subscribers defined by a VC.

A simplified block diagram of the basic switch or switching node structure 22 usable for each EFPS or TFPS is shown in FIG. 2. The different functionalities of the switch 22 are accommodated by the manner in which connections are made in the Switching Fabric Subsystem (SFS) 24, as will be described presently. SFS 24, Subscriber Line Subsystem(s) (SLS) 25 and Trunk Line Subsystem(s) (TLS) 26 provide the major infrastructure of the switch. SLS 25 includes one or more Universal Control Units (UCU) 27 each of which is associated with one or more Subscriber Processing Units (SPU) 28, and if desired, a Port Multiplexer/Controller (PMC) (not shown). The SPU(s) 28 and associated UCU 27 communicate via a system peripheral bus 30. The PMC may be used to provide extended multiplexed access and control to SFS 24.

Each SLS 25 supports system protocols, provides access to network subscribers (which, for example, may be individual telephone, TI trunk, PBX signal, computer and/or other devices, lines or signals) on lines such as 31, 32, 33 and 34 at the endpoint where switch 22 is located (if the switch is used in the EFPS mode or functionality), and provides the interface to the SFS 24. The SPU 28 is implemented to provide access, support and control for the designated category of each of the subscriber lines, maintain intelligent interface to the associated UCU to provide flow control and network management functions bidirectionally on the peripheral bus, and perform all necessary native protocol emulation.

The UCU 27 is implemented to provide FPS internal protocol support in either of two modes, a tandem mode or a stand-alone mode. In the tandem mode, two UCUs share responsibility for configurable frame formatting and dispatching. Toward that end, the UCU in the SLS 25 sends subscriber data streams to an associated UCU in the TLS 26 for composition of the frame payload. In the stand-alone mode, the UCU in the SLS handles the entire process. In a sense, the UCU acts as a concentrator, receiving data from the various subscribers via the SPUs, concentrating the data, providing the necessary levels of functionality, and presents the data to the switching fabric (SFS) for routing to a TLS and subsequent transmission to the external world.

TLS 26 also has UCU(s) 36, which provides the functionality described above for the SLS/UCU(s), and Trunk Processing Unit(s) (TPU) 37, which provides access, support and control for the FPS trunk lines such as 38 and 39, and a physical interface to the associated UCU for frame transmission, error detection and correction, and synchronization. For example, the data from the SLS 25 is received at the TLS 26 after traversing the switching fabric, is collected by the UCU 36, composed in the frame payload and presented to the TPU 37 for transmission to the next node.

Several different connection scenarios—SLS to SLS, or SLS to TLS, or TLS to TLS—in the switching fabric are available (shown in dotted lines in FIG. 2) according to the desired use of the switch. The connection of TLS to TLS provides transit switch (TFPS) functionality. An SLS to TLS connection provides endpoint node (EFPS) functionality from the subscriber to the trunk; and SLS to SLS connection provides functionality internal to the node from one subscriber to another subscriber.

In the exemplary embodiment each SLS 25 and TLS 26 supports T1/T3 interfaces because this BW range is more suited to effective implementation of the composite frame, but other interfaces are not precluded. At Tl/T3, the data stream at the SFS should be$\leqq$1.544 mbps (2.048 mbps in European standard).

It is desirable at times to refer to "source" and "destination" or to use other, but analogous, terms to identify the two sides of a logical connection—whether in reference to subscriber connections (VCs) or EFPS connections (VCPs). The two sides of a connection will also be referred to sometimes herein as the local side and the remote side. At times, the remote side may be the destination side; and at other times, the remote side may be the source side. In the architecture for VCPs according to the present invention, however, the source side of the VCP connection is determined (i.e., designated) at the time that the particular VCP is created.

For example, referring to FIG. 3, a trunk line subsystem (TLS) 40 associated with EFPS 41 is implemented and organized to recognize the need to build a VCP upon receipt of a number of subscriber connection (VC) requests destined for the same endpoint EFPS 43, from subscriber line subsystems (SLSs) 44. At that point, TLS 40 initiates a VCP call request (CR) and sends it to the "destination" TLS 45 associated with EFPS 43. If TLS 45 responds to the CR with a call accept (CA), which will depend upon customary considerations for establishing a call, a VCP is established between the two endpoint EFPSs. Because the CR originated from the EFPS 41 side of the connection, that side is thereafter referred to as the "source" side of the VCP, and the other side—the EFPS 43 side—is termed the "destination" side, of this particular VCP.

The concept of source and destination sides of the connection is useful for a variety of reasons. For example, if the connection of interest were to broken inadvertently, such as because of a link failure, or if re-synchronization (discussed below) of the connection (or the entire network) were required, it is desirable—indeed essential—that one side should be passive and the other side active. Accordingly, although the source side is somewhat arbitrarily designated, once that designation is made the source side becomes and remains responsible for all synchronization type activities. Hence, whenever an end to end network re-synchronization activity is required, the source side of the particular VCP connection performs that activity.

The relationship of VCs to VCPs is illustrated in the block diagram of FIG. 4. FPS network 50 is an integrated services network, and includes a multiplicity of EFPSs 52, 53, 54 and 55 each of which has a plurality of subscribers associated with it. Each of the EFPSs has a TLS for each of the trunk lines (such as 57, 58 and 59 for EFPS 54) connected to that EFPS, and SLSs for the subscribers (such as 61, 62 and 63 for EFPS 54) associated with that EFPS, as described above for FIG. 2.

EFPS 53 has subscribers 66, 67 and 68 which have initiated call requests to subscribers at other endpoints of network 50, sufficient to justify the establishment of VCPs (EFPS connections) for the VCs (subscriber connections). For example, subscriber 66 associated with the latter EFPS has a VC with subscriber 71 of EFPS 52, and another VC exists with that same subscriber 71 and another subscriber at EFPS 53, resulting in the establishment of VCP 80. Subscriber 66 also has one VC with a subscriber (74) of EFPS 55, and other subscribers (67 and 68) of the same originating EFPS (53) also have VCs with subscribers (74, 75 and 76) of the same destination EFPS (55). These VCs sharing a common source/destination EFPS pair are multiplexed onto a single VCP 82, which traverses TFPS 83 (the only VCP transit hop in this example).

Although the VCP concept itself is not new, the concept is implemented in a unique and different manner according to the present invention, with considerable benefits accruing from establishing VCPs across the integrated services network as a result. The VCP is physically represented by the composite frame which can carry many different traffic component types. Traffic is allowed to flow in the form of numerous subscriber connections (VCs) occupying respective channels in composite data frames between many pairs of source and destination EFPSs. Most of the benefits of the VCP are enjoyed at the transit nodes (TFPSs), including, for example, a many-fold reduction in call setups and call clearings, the specific number depending on the ratio of packet processing for call setup/teardown to packet processing for data for transit hops (i.e., between transit nodes) in the network. The VCP connection remains in place for traffic—subscriber connections—between the same pair of EFPSs, and thereby eliminates the need for the TFPSs along the VCP to continually setup and tear down connections as VCs that connect individual subscribers are established and terminated. A beneficial fallout of this reduction in processing is that the TFPS need not perform routing, control block allocation/linkage/release, or recording/checking of state information, on a per call basis.

Another advantage is that assuming an average of, say, ten VCs multiplexed on a single VCP (which is not an inordinate number in this scheme), an order of magnitude reduction in memory requirements for VC control would be enjoyed at each TFPS. A further advantage is that network processing and delay for link and node failures are reduced by an order of magnitude at the transit nodes by virtue of their need to perform reconnect processing only for their respective VCPs, rather than for every VC that traverses the node.

Although the source and destination EFPSs do not share these benefits directly with the TFPSs, they, too, perform less packet processing than would be the case without VCPs. This is because there is no inherent protocol conversion or packet reformatting associated with the VCP scheme.

The advantages of establishing VCPs increase with increases in the traffic load and with decreases in traffic fan-out (i.e., increases in concentration). As more VCs share source/destination EFPS pairs, more VCs can be multiplexed per VCP. Hence, the greatest benefits of VCPs are enjoyed on paths of highest traffic concentration between endpoint nodes. As will be described presently, the present invention provides techniques and implementations by which the VCPs may be anchored adaptively to different switching nodes of the integrated services network.

In fast packet switching networks, a conflict typically exists between the requirements of low delay, high throughput, and maximum bandwidth utilization. It becomes highly desirable to develop a data frame or packet format which will resolve the conflict, but the prior art schemes have not proved successful in that regard. For example, if the frame is small, with a ratio of payload size to header size approaching unity, it has a good packetization delay in that frames are launched relatively quickly, because the payload channel is filled rapidly with the subscriber data stream. On the other hand, each frame has as much as 50% of its bandwidth devoted (i.e., locked) to header information, and the balance devoted to the payload, which constitutes very poor bandwidth utilization. Also, this type of frame format results in a low effective packet switch throughput because each frame contains a relatively small amount of payload data and a relatively large amount of header or control information. A large number of frames must be switched to transfer any significant amount of data since the switch processing time and complexity is a function of the size of the header, and is not favorably affected by the small size of the payload.

If the frame is too large, in the sense that the ratio of payload size to header size is substantial, packetization delay is poor in low speed applications because the payload channel takes an inordinate amount of time to fill. On the other hand, if such a frame format is utilized in short burst applications, the frames will be launched with little delay but with a relatively large amount of unoccupied space in the payload channel, i.e., the frame has a great amount of its bandwidth devoted to unused payload, and thus represents poor bandwidth utilization. The foregoing and other drawbacks are found in the fixed size single subscriber payload frame (or cell) proposed by the CCITT ATM standard. The current ATM standard is a cell having a payload size of 48 bytes and a header size of five bytes.

The use of a variable size single subscriber payload frame also produces a poor result. The X0.25 standard is such a frame format, with payload size ranging up to 4,096 bytes, in addition to header, and suffers from poor bandwidth utilization and switch throughput for short burst data for the same reasons as those applicable to small fixed size payloads. Also, more complex algorithms are required to handle the worst case delay and packet jitter experienced for isochronous services in a multimedia network.

The configurable frame format of the system and method of the present invention is a distinct improvement over the fixed size and variable size payload frames of the prior art. A preferred method and embodiment utilizing such a format will now be described for a suitable high speed data (fast packet) network.

Only the subscriber data streams at an EFPS which are intended to be sent to another, common EFPS of the ISN network (or of another network linked to that network) may be combined or assembled in the same composite data frame. It will be understood that in this context, the term "data" is used in a broad sense, encompassing data, voice, video and any other traffic component types. For example, referring again to network 10 of FIG. 1, if several subscribers at endpoint A are sending data to subscribers at endpoint B during a given time interval, the data streams of those endpoint A subscribers may be combined in composite data frames by the EFPS at endpoint A. These frames are then launched to be transported through the network to endpoint B for appropriate distribution to the respective subscriber destinations.

In keeping with the previous discussion of source and destination labels, although certain subscribers, switches or endpoints may be variously described as entry, source or origination, or as exit or destination, or by analogous terminology, any of them may (and typically will) act as both a source and a destination, in the customary sense, in any communication session(s) across the network. In other words, data may flow in both directions in any given VC over the course of the connection between the two subscribers. However, the previously mentioned convention of "source" and "destination" designations for purposes of the architecture of the system continues to apply.

Referring to FIG. 5, an exemplary composite data frame according to the invention has a fixed payload size but is composed in a way to accommodate a plurality of traffic component types. Exemplary composite data frame 90 is 192 bytes in length, including a header 92, payload 93 and frame check sequence (FCS) 94. Payload 93 contains the information to be communicated to subscribers at the destination endpoint. The payload is divided into traffic component slots (referred to herein as T-slots) 96, 97 and 98, in this example, with grouping according to traffic component type (such as voice and data here, although other components such as video may also be included in a separate T-slot of the frame). That is, each T-slot is dedicated to subscriber connections of the same traffic component type. In turn, each T-slot is subdivided into multiple channels, such as the three channels 99, 100 and 101 for T-slot 96.

According to the preferred embodiment and method of the invention, each channel of a T-slot is allocated when a subscriber connection is requested, and remains dedicated to that single active subscriber connection (VC) for the life of the connection. The channel is released only when the connection is terminated. An alternative scheme is to configure and reconfigure the data frame with each start and end of a burst, and to allocate the channel only for the duration of the burst. However, the latter scheme requires more configuring and reconfiguring of the frames and more overhead and control information to be transported through the network than the duration of connection technique. Consequently, the preferred technique provides a relative reduction in packet configuration/reconfiguration and bandwidth overhead requirements. An additional advantage of the duration of connection technique, namely, no loss of data, is realized from the frame compression mechanism which is another aspect of the invention, to be described presently.

In the illustrative example of FIG. 5, payload 93 of the composite data frame is assembled from three different traffic component types—64 kilobit (kb) X0.25 data, adaptive pulse code modulation (ADPCM) voice, and 9.6 kb SDLC (Synchronous Data Link Control) data—which are consequently grouped in three T-slots. The number of T-slots in the payload of the frame and the number of channels per T-slot may be more or less than are shown in this example, subject to the limitation of frame length. With the composite framing technique of the present invention, each frame may be configured and reconfigured according to the transmission needs and traffic component types of the various subscribers at each EFPS.

According to a further feature of the system and method of the invention, the channel sizes in each composite data frame (such as that of FIG. 5) are T-slot specific, and this condition exists network-wide, which has the advantage of eliminating the need for control information to specify channel size. That is, all channels of a specific T-slot type (such as X0.25 data) in all frames on all VCPs in the ISN network are of equal size, and, because this is a known quantity, the overhead (bandwidth) which would otherwise be required to designate channel length is eliminated from the frame. Selection of channel length is made statically by the network administrator, according to the nature of the respective traffic component. In the example of FIG. 5, the selection of channel sizes for the three T-slots is based on a single traffic component type attribute or characteristic, namely, the ratio of "subscriber line rate" to "packetization delay". However, in practice many other attributes (such as activity level, burst size, and so forth) may be considered.

Another example of an important attribute for packet switched services in the context of the present invention is average packet length, because it is desirable to put an entire packet into a single channel of the composite data frame. If that is not feasible, it is desirable to minimize the amount of chopping or dissection of the data. For example, in an extreme case one packet may be 128 bytes long and another may be 10,000 bytes long. The development of an appropriate compromise is relegated to selecting the optimum increment for dividing up the packet. Another important attribute for isochronous and circuit switched services is sensitivity to delay, which also dictates buffering and packetization delay. For example, if the buffer is very small, it is necessary to packetize and ship out the data as soon as it is accumulated. In those circumstances, the channel size should be selected to be as small as practicable, because the larger the channel size the more delay is introduced into the system. On the other hand, in the case of a variable bit rate source, there is a probability of overflow of the internal buffers. If the channel is too small, the probability of buffer overflow is high, and with each overflow some data is lost. It is necessary in those circumstances to increase the buffer or the channel size. In the case of the channel the frame compression mechanism of the present invention is available, but in the case of the buffer there is underutilization which, nevertheless, is acceptable with a variable bit rate source because although infrequent, the buffer can be filled quickly when high speed (bit rate) data is being received at the buffer. For the variable bit rate source, one attribute is the probability of overflowing the buffer or channel.

According to an aspect of the invention, each VCP of the ISN network has an autonomous composite data frame format which is defined and managed by the anchoring EFPSs at the source side and the destination side of the respective VCP. That is, each VCP has only one frame format associated with it at any given time. The frame format may be set permanently at the time that the frame is configured; or it may be reconfigured dynamically by either of the EFPSs anchoring the VCP upon detection of a subscriber connection (VC) request or release, which is the preferred scheme. The VC request (for addition of a channel) or release (by deletion of a channel) is contained in a frame reconfiguration request (FRR, which will be discussed in greater detail presently) issued by the EFPS which is servicing the active subscriber (i.e., the subscriber desiring to add or to release a channel). An FRR is never issued by a TFPS, for reasons which will become apparent in the subsequent description of this feature herein.

The composite data frame payload format, to the extent that it is reconfigured (and only upon such reconfiguration), is set forth in a delta change template (i.e., signifying only the changes from the prior format) which is conveyed within a control frame by the local anchor EFPS to each of the transit nodes (TFPSs) of the VCP and to the remote anchor EFPS. This "payload format template" is stored at each of those nodes.

The composite data frames are composed and decomposed from and to the data streams of the respective subscribers, only by the local and remote EFPSs constituting the anchor nodes for the VCP, and never by the TFPSs along the VCP. Each composite data frame has a format exemplified by FIG. 5, discussed above. The frame format is transparent on the transit nodes except during periods of output link congestion

As described above, during frame composition a dedicated fixed size channel is allocated in the composite data frame payload for each subscriber connection (VC) multiplexed onto the VCP. The channels are grouped together in T-slots (traffic component types) in the payload. Channel sizes may vary from T-slot type to T-slot type, but are a predefined fixed size for any given T-slot type, network-wide. Thus, in FIG. 5, for example, the X0.25 T-slot 96 has three channels 99, 100, 101 of equal size, each having a 64 bit length. The PCM voice T-slot 97 has two channels of equal size, each having a 32 bit length. And the SDLC data T-slot 98 has two channels of equal (16 bit) size.

Because the channel size for each particular T-slot type is the same on a network-wide basis, and is known throughout the network, it need not by carried within each frame as part of the control information, thereby conserving bandwidth. Only the T-slot type, its position in the frame, and number of channels need be communicated in order to provide a complete picture for decomposition of the frame at the destination VCP. The principal criteria for setting channel size are (1) subscriber interface speed and (2) activity level of the T-slot type. In the preferred embodiment, the header size is a minimum of 5 bytes in a composite data frame maximum length of 192 bytes.

The composite data frame format in general, and the payload format in particular, provide a number of benefits when compared to other payload formats such as fixed size single subscriber payload format and variable size single subscriber payload format. One benefit is related to the traditional capability of network subscribers to optimize resources by performing multi-subscriber multiplexing outside the network, for example, by installing a control unit between the network and a cluster of 3270 terminals. This type of optimization is extended by the network, by means of the configurable frame format, across dissimilar devices and traffic components provided that they share the same source/destination EFPS pair. Moreover, the shared payload scheme of the invention possesses advantages over traditional packet network multiplexing schemes. One such advantage is the capability to improve bandwidth utilization by providing a larger payload size to header size ratio without the normal packetization delay. Another is that the effective switch throughput is increased by carrying the subscriber data streams with larger payloads, in that for a given volume of subscriber data, throughput is more a factor of frame count than frame size. For example, considerably more processing is required for one thousand 100-byte packets than for one hundred 1,000-byte packets.

By configuring the VCP composite data frames with many very small channels (e.g., 2 bytes), to create the illusion of a continuous bit stream with no packet jitter, the packet switched network assumes the circuit switched network characteristics of enhanced transmission quality. Thereby, the configurable composite data frame may be used as part of a circuit/packet switch hybrid product. The frame may also be configured to accommodate ATM schemes and large payload fast packet switch schemes.

The composite data frame header fields are functionally analogous to header fields in ATM cells and LAPD (link access procedure) frames, the major differences being in format to reflect the requirements of an end-to-end protocol in contrast to the interface protocols addressed by standards of the latter two packet types. FIG. 6 presents a simplified comparison of the three packet types, with the composite data frame for the preferred embodiment of the present invention shown in part (a), and the ATM cell and LAPD frame of the prior art shown in parts (b) and (c), respectively.

Referring to FIG. 6(*a*), the FLAG field is a frame delimiting flag required for synchronization because of the variable frame lengths. The trailer flag of a preceding frame may act as the preamble flag of the next frame. The PT (payload type) field identifies the frame type, i.e., a data frame or one of the defined control frames. This is unlike the ATM cell header PT field (FIG. 6(*b*)) in the following respects: (1) in the data frame of the present invention, the PT field precedes the VPI (virtual path ID) field because both will be considered simultaneously for switching purposes and the PT field has higher precedence; and (*b*) the PT field is larger than the ATM cell PT field because of the added control frame types required by the internal protocol.

The VPI field of the composite data frame has local input/output link significance. Two bytes allow for up to 64,000 possible VCPs to traverse a given link—a worst case which allows for maximum utilization of a T3 link with an average VCP activity level of only 700 bits per second (45 mbps/64 k VCPs=700 bps per VCP). The VER (version number of the data frame) field provides the version number which is needed for synchronization because the payload format is dynamically modifiable via VCP control frames. The VCP source and destination anchors (EFPSs) have a one bit version number to toggle for this purpose.

A significant feature of the composite data frame of the present invention which serves, in part, to distinguish it from frames, packets or cells of the prior art (whether those shown in FIG. 6(*b*) and (*c*) or otherwise), is the PFC field. In the composite data frame of FIG. 6(*a*) this field provides prioritized flow control (PFC). It will be observed that the ATM cell has a PFC (priority fairness control) field as the initial field in its header which, however, is understood thus far to be undefined by CCITT. According to the present invention, prioritized flow control as encompassed by the PFC field of composite data frame header is used for the purpose of providing frame compression, to conserve bandwidth and optimize bandwidth usage. This technique will be discussed in detail below, but for the present it is only necessary to briefly mention some aspects of the PFC field.

The PFC field of FIG. 6(*a*) has three types of bits for each T-slot: a single A bit for flow control, and a single B bit and a number of C bits equal to the number of channels in the T-slot, the C and B bits being referred to herein as "presence" bits. The A bit is used by any TFPS or anchor EFPS of the VCP to inform the TLS or SLS therein of the requirement and severity of flow control. For example, A=1 indicates that flow control is required; whereas A=0 indicates that no flow control is required. The B bit is used by the VCP anchors to indicate an absence of all channels associated with this T-slot in the composite data frame (i.e., that the frame is compressed). For example, if B=1 the T-slot identified by that bit is present, and if B=0 the T-slot is not present (i.e., all of the channels associated with that T-slot are absent). The C bits are associated with the respective channels of the T-slot, one bit per channel, and are used by the VCP anchors to indicate frame compression (absence of the channel) attributable to either subscriber inactivity or flow control. For example, if C=1 the channel identified by that bit in that specific T-slot is present, and if C=0 that channel is not present in the T-slot.

It follows that if, for example, a frame has three T-slots and four channels in each T-slot, the PFC field would contain three A bits, three B bits and twelve C bits (positioned in the sequence A, B, C for each T-slot as shown in FIG. 6(*a*)). If B=0 for a particular T-slot, none of its associated C bits are carried in the field; i.e., if a T-slot is not present, none of the channels associated with that T-slot is present in the data frame, and no bandwidth need be allocated in the payload for the corresponding channels.

Thus, the PFC bits provide a complete picture of the state of flow control required for the respective T-slot at the network nodes traversed by the particular VCP. This control information is built into the frame header by the source EFPS during frame composition, and neither the transit switches on that VCP nor the destination EFPS for that frame can modify the B and C bits. However, each FPS (TFPS or EFPS) can change the value of the A bit to trigger flow control, when necessary. In essence, the B and C bits are used during frame decomposition to communicate whether there is frame compression as a result of the absence of the associated T-slot or a channel thereof, either because of inactivity of the respective subscriber(s) or because of flow control.

The PFC field of the composite data frame header is immediately followed by the payload, which is followed by an FCS (frame check sequence). Just as in the case of the FCS of the LAPD frame of FIG. 6(*c*), this FCS applies to the entire frame. The ATM HEC (header error check) field of FIG. 6(*b*) is analogous to a frame check sequence, but applies only to the header. Application of a check sequence to the entire frame is desirable because it provides an additional level of network integrity through payload error detection. In this respect, it is also noteworthy that ATM is targeted for 100% fiber optic networks which have very low bit error rates. The composite data frame FCS in the preferred embodiment of the invention is only one byte long compared to the two byte LAPD FCS, but nevertheless provides a comparatively high level of integrity because of the maximum frame length of 192 bytes versus 4,096 bytes for the LAPD frame.

For ATM, the cell (packet) payload is a single channel of fixed size as shown in FIG. 6 (*b*), and successive cells are launched asynchronously only after the subscriber data stream has filled the channel. A feature of the system and method of the present invention is that the composite data frames are launched synchronously. This is a departure from conventional principles of circuit switched networks, which are invariably synchronous, and packet switched networks, which are invariably asynchronous. The synchronous frame launching further enhances the performance characteristics of the packet switched network (beyond enhancement obtained from other features of the invention, such as the use of a frame with many relatively small channels), to provide improved transmission quality comparable to that of circuit switched networks, while retaining the bandwidth savings advantage of packet switching. Isochronous services such as voice and video are provided with near circuit switching quality jitter and delay characteristics by employing this combination of fixed short time interval packetization and synchronous frame launching onto the ISN network. Synchronous frame launching also permits fixed size channels on a "per T-slot" basis, and consequent elimination of any need to transmit channel length control information for purposes of frame decomposition.

A simplified block diagrammatic representation of a VCP with synchronous frame launching is shown in FIG. 7. EFPSs **111** and **112** anchoring VCP **113** are responsible for frame composition/decomposition at their respective endpoints. Each of those EFPSs composes and transmits a composite data frame (x and y, respectively) toward the remote EFPS anchor (**112** and **111** as the case may be, in this example) at a fixed time interval relative to the last frame launching and to the next frame launching. This time interval, which is synonymous with packetization delay, is configurable on a network-wide basis, and may be, say, one or two milliseconds (ms) in length. In the preferred embodiment, a launch interval of one ms is used throughout the network for synchronous frame launching. As a consequence of this network-wide synchronous frame launching, the launch interval information is known at each switch along the VCP, whether TFPS or EFPS. The only additional information which is required for frame decomposition at the remote EFPS is the number of channels allocated in the frame and the traffic component type of each channel. This not only simplifies the decomposition process, but, by reducing the amount of control information which must be carried in the frame, results in a significant saving of bandwidth.

An EFPS, or more precisely the point of multiplexing within the EFPS, may anchor more than one VCP. However, for each VCP established as a logical connection between a pair of EFPSs, there is a one-to-one coupling with the composite data frame transported on it. In the example of FIG. 7, EFPS **111** launches composite data frame x+1 one ms after frame x was launched. Similarly, the other anchor for this VCP, EFPS **112** launches composite data frame y+1 one ms after frame y. The channels in each frame payload contain subscriber data which was accumulated during the one ms since the last frame was launched from that EFPS. For example, 8 bytes is the maximum amount of data which can be accumulated for a 64 kb PCM voice subscriber in a one ms frame launch network (64 kb/1000=64 bits=8 bytes). A two ms frame launch network would require double the optimal channel size for a given T-slot (e.g., 64 kb PCM voice T-slot can fill a 16-byte channel in two ms). Thus, the frame launch time should be optimized for the T-slot types intended to be transported by the particular ISN network.

A decision as to whether to send a partially filled channel or to delay until the next frame launch will depend on the attributes of the T-slot type to which that decision applies. In any event, the PFC field in the header of each composite data frame will signify any channel omission, and thus frame compression that may be present, with an elimination of otherwise wasted bandwidth.

The composite data frame format scheme and synchronous frame launch scheme have a certain dependency, and consequently, overlapping benefits such as virtual elimination of one of the two major "quality of service" differences between packet and circuit switched networks, namely, packetization delay. (The other, bandwidth reservation, will be discussed presently). Various other benefits accrue in the packet switched ISN from the use of synchronous frame launching. One is a substantial reduction in packet jitter for isochronous services subscribers. Major contributors of packet jitter and packetization delay in prior art techniques are (i) variable sized packets and/or indeterminate "packet launch times" attempting to fully utilize a fixed size frame, and (ii) long periods of subscriber data buffering in an attempt to improve header size/payload size ratio. Among solutions which have been proposed in the prior art to alleviate these problems are the use of (i) partially filled ATM cells (less than 50% bandwidth utilization for PCM voice); (ii) echo cancellation equipment at each subscriber port; and/or (iii) packet payload sharing among several subscribers. The present invention utilizes a variation of the latter technique in one of its aspects, in a manner which is a considerable improvement from standpoints of simplicity and effectiveness, over the specific solutions heretofore proposed.

Among other benefits obtained from the use of synchronous frame launching in the ISN FPS network of the invention are a reduced occurrence of partially empty payloads as a result of smaller payload channels, with an attendant improvement in bandwidth utilization; and the capability to accurately calculate worst case traffic spikes at the transit nodes as a result of a fixed frame launch period in conjunction with a fixed channel size, which allows better management of bandwidth reservation and allocation. Also, network frame loss has significantly less impact on isochronous services subscribers, because of the smaller channel sizes.

The advantages of synchronous frame launching in the ISN network are greatest when multiple subscribers actively share each VCP, so that the payload channel count (i.e., payload size) is sufficiently large to maintain a good ratio of payload size to header size while using a relatively short launch time interval. An example of a payload size resulting from six "highly active" 64 kb subscribers is illustrated in FIG. 8. The network-wide frame launch time is one ms in this example. T-slot **1** is 64 kb transparent circuit mode voice, and has four channels (one for each of these voice subscribers). T-slot **2** is composed of a 64 kb subscriber frame relayed data stream, with two channels (one for each of those data subscribers). It should be noted that, in this example, although all of the subscribers' data streams have the same speed (64 kb), and the channel size for both of the two T-slots is the same (8 bytes), the payload contains two different traffic component types, i.e., (1) voice and (2) frame relayed data. The VCP on which this composite data frame is transported is capable of fully utilizing the frame with an overall header and payload size equal to that of an ATM cell, and a ratio of payload size to header size of approximately ten, while complying with only a one ms packetization delay.

In systems of the present invention, multi-media traffic integration with predictable service quality and transmission facility economy is provided with minimal network management system (NMS) configuration and intervention activity. Bandwidth management intelligence is provided by the FPS network itself. The NMS parameters include network-wide configurable parameters and transmission link specific parameters. The network-wide parameters, which permit traffic-driven specification of the network, complete control of the trade-off between quality of service (on a per-T-slot basis) and transmission facility utilization, and prioritization by T-slot, include those listed in Table I, below. T-slot type channel sizes are indicated in Table II. Rather than designating a particular traffic component with a priority level for all of its attributes, it may be desirable, and is preferred in the embodiment and method of the invention described herein, that a priority level be assigned for each different attribute of the traffic component (including attributes such as delay sensitivity, loss tolerance, activity level, burst size, average packet length, probability of buffer or channel overflow, etc.). Priority is used during flow control/congestion situations to determine which traffic components should be call blocked and/or put in a degraded service mode.

The NMS transmission link specific parameters allow the NMS to divide the network into autonomous service points/regions, and to control at each point (during peak traffic loads) the quality of service on a per T-slot type basis. These parameters include link utilization threshold and T-slot bandwidth allocation profile, the latter including T-slot minimum guaranteed bandwidth, T-slot maximum allowable bandwidth, and call block threshold, listed in Tables III and IV, respectively, below. All values set forth in these tables are presented for the sake of example only.

TABLE I

Name	Range	Default	Comment/Default Rationale
max frame size	16–192 (bytes)	192	Preferred FPS network BW manager for optimizing
frame launch interval	1-5 ms	1 ms	Shortest interval permitting adequately large channels to be fully utilized[1]
max T-slots per frame	1-5	4[2]	More would lead to excessively large PFC field, and cumbersome congestion processing on transit nodes
max channels per frame	1-30	20	Default value based on T1 & T3 trunks, w/low-end subscriber rate of 64 kb (i.e. channel size 6 to 8 bytes)
min channel size	2-187 (bytes)	2	Allows full channel utilization down to 16 kb subscriber w/o "sub-multiplexing outside network
max channel size	2-187 (bytes)	187	Good compromise between memory mgmt limitations & industry optimal size?[3]
max T-slot types ntwk-wide	1-32	16	Consistent with ATM standard of 16 priority levels

TABLE I-continued

Name	Range	Default	Comment/Default Rationale
T-slot type channel size			**see Table II**

[1] Value directly impacts channel size of each T-slot type. As launch time increases. channel size can increase with full payload utilization. The trade-off is increased packetization delay.

[2] If a source/destination EFPS pair must carry more T-slots or channels simultaneously, addl. VCPs may be set up between the pair.

[3] Small enough to manage max packet size in a single buffer throughout system (incl. switch fabric). European group prefers 32-byte "channels", while other extreme advocates 600-byte "channels"

TABLE II

T-SLOT TYPE CHANNEL SIZES[4]

T-slot (priority)[5]	T-slot Description	Channel Size (No. of Bytes)
0	low scan video	75
1	ADPCM voice (32 kb)	4
2	64 kb packet data	8
.		
.		
.		
31	—	—

[4] T-slot descriptions and channel size values are exemplary only.

[5] Priority is used during flow control/congestion situations to determine which traffic components should be call blocked and/or put in a degraded service mode.

TABLE III

LINK UTILIZATION THRESHOLD

Level	Threshold
1	50%
2	65%
3	80%
4	90%

Each increasing threshold affects one or a range of T-slot types with higher priority. "Threshold" is a measure of actual (current) bit usage as a percentage of link bit rate (i.e., 750 kbps actual on 1.5 mbps link = 50% threshold. The effective bit rate (amount of subscriber data transmitted) is less than or equal to actual bit usage, because of header fields and partially empty payloads.

TABLE IV

T-SLOT BANDWIDTH ALLOCATION PROFILE TABLE

T-slot Type	Min Guaranteed BW (as % of link bit rate)	Max Allowable BW	Call Block Threshold (Link Utilization Threshold)
0	10%	50%	4
1	20%	50%	1
2	10%	50%	1
3	30%	30%	2
.			
.			
.			
31	—	—	—
	Total < max link utilization threshold value		

Notes to Table IV:

A T-slot type may exceed its maximum guaranteed bandwidth during low traffic conditions, but the FPS can seize back (by flow control and call blocking, to be explained in more detail below) all of the bandwidth exceeding the minimum guaranteed level. Seizing proceeds on a schedule of lowest priority first.

By setting values for min guaranteed BW and max allowable BW close together, the NMS can ensure the quality and predictability of service to the T-slot because this assures that BW will not be seized away and redistributed to another T-slot type. Setting the values far apart allows delay-insensitive T-slot types to far exceed their min guaranteed BW, with the risk of flow control delay if traffic load increases from other T-slots.

Call block threshold is a measurement of total BW in use by all T-slot types on the link. It is not enforced (not relevant) unless the request for BW is by a T-slot type whose minimum guaranteed BW is already exceeded. The purpose of this parameter is to allow the NMS to reserve BW for high priority T-slot types, even when they are inactive, by blocking lower priority T-slots. Such configuration may be used to prevent BW allocation policy thrashing - where BW is constantly redistributed among several contending T-slot types.

Another key aspect of systems according to the invention is that within the framework of the NMS configuration parameters, the ISN FPS network policies maximize effective link throughput by providing on-demand bandwidth distribution/redistribution among T-slot types while generating minimal control information traffic. Toward that end, the SPU of the SLS (described in conjunction with FIG. 2 above) supports the native protocol, and reserves bandwidth on the VCP by requesting a channel of the appropriate T-slot type from the local VCP anchor (EFPS). In a request for bandwidth, the SPU requests or releases the associated VCP channel upon receipt of a connection request (call request) or connection release (call release) from the subscriber to which that channel is allocated. In the preferred embodiment, for at least some T-slot types with a high volume of short duration connections to the same destination, the SPU will hold a pool of channels on the VCP and request/release channels on the basis of the current number of active calls on those channels within a predetermined high/low threshold, rather than on a per subscriber call basis. This type of multiplexing is transparent to the system protocol, which views each channel as a single, indivisible subscriber entity.

Referring back to FIG. 2, the SPU such as 28 selects the desired VCP for the request/release based on a suitable FPS routing algorithm stored in the switch in which the SPU is located. If the current VCP has reached its maximum size, or if there is no VCP to the desired destination EFPS, a new VCP is sought to be established by the local anchor EFPS. The SPU channel requests are sent to the TLS 26 anchoring the VCP. Channel requests/releases are initiated only at the VCP anchor EFPSs, but TFPSs along the VCP may reject requests based on link traffic conditions and based on the NMS transmission link specific parameters described in connection with Tables III and IV above.

In the processing of bandwidth requests, for each channel request/release by the SPU, the associated local VCP anchor EFPS builds and launches a composite data frame reconfiguration request (FRR) control frame. When multiple reconfiguration requests for a VCP are received by the TLS during a single frame launch period, all of those requests are combined into a single FRR control frame. In response to receipt of the FRR control frame (or simply, FRR) each successive TFPS on the VCP and the remote anchor EFPS, in turn, will either (i) reserve the requested bandwidth and relay the FRR to the next node (transit or endpoint, as the case may be), or (ii) send a rejection (i.e., refusal to reserve the requested bandwidth) back to the originating anchor, depending on traffic conditions and the NMS parameters then existing at that node.

The volume of such FRR traffic on a VCP is a function of the number of T-slot types on the VCP and the number of channels per T-slot. Table V below illustrates two exemplary VCP traffic profiles and the corresponding amount of FRR traffic generated thereby (in number of composite data frames per FRR control frame).

TABLE V[6,7]

Example 1:	
VCP traffic =	10 PCM voice connections
Result:	avg time between reconfigurations = 6 secs = 1 per 6,000 data frames (steady state requires avg turnover of 5 callers every 60 secs) composite data frame payload size = 80 bytes (ten 8-byte channels)
Example 2:	
VCP traffic =	5 facsimile connections and 10 E-mail connections
Result:	avg time between reconfigurations = 4 secs = 1 per 4,000 data frames

[6] Following assumptions apply:
each call requires two configurations (channel request/release)
network frame launch period = 1 ms
PCM voice average call duration = 120 seconds
Fax call duration = 60 seconds
E-mail call duration = 240 seconds

[7] Examples given are for steady state traffic loads. Manner in which loads are achieved is irrelevant to illustrating frequency of composite data frame reconfiguration. (steady state requires avg turnover of 5 fax and 2.5 E-mail callers every 60 secs)

In processing an FRR request rejection (call blocking), the SPU is given the capability to reroute the requested channel or bandwidth on the existing VCP by means of a routing algorithm, or to request that a new VCP be established by the local anchor EFPS, in the same way as discussed above.

Each node on a VCP (both the anchoring EFPSs and the TFPSs on the VCP) individually makes a determination of whether to grant an incoming FRR. This is accomplished by analysis at the respective node of (i) the current bandwidth usage profile of the local output trunk line; (ii) the values of the NMS bandwidth configuration parameters; and (iii) the requested redistribution of bandwidth. Such analysis is performed only in response to an FRR requesting additional bandwidth, i.e., a T-slot channel request. Because principles of bandwidth conservation and maximum utilization are paramount in the system, FRR requests to release bandwidth, i.e., T-slot channel releases, are responded to by an automatic grant of the FRR.

The bandwidth allocation rules (algorithm) utilized in the presently preferred embodiment of the system and method of the invention are as follows:

A request for bandwidth is granted to an SPU only if each node on the VCP "agrees to it". Thus, both of the VCP anchor EFPSs and all of the TFPSs along the VCP have the ability to block the request.

A VCP node will grant a request for bandwidth if the local output link of the VCP meets the following criteria/algorithm[8]:

1) total BW in use<maximum utilization threshold;
2) requested T-slot type BW usage<max allowable BW usage for this T-slot type; and
3) requested T-slot type current BW usage<min guaranteed BW for this T-slot type, or total reserved BW on ink<requested T-slot type call block threshold.

FIG. 9 illustrates examples of bandwidth allocation requests (FRRs) under various traffic conditions, i.e., BW grant/reject scenarios. Assumptions used for each case are as follows:

The link threshold utilization table is

level	threshold
1	50%
2	65%
3	80%

The link specific parameters for the T-slot type of the channel requested are:

[8] The algorithm uses NMS transmission link specific parameters, discussed above. It should be observed that the bandwidth rules are based on the link profile which is an aggregate of all VCPs traversing the link—in contrast to bandwidth requests from an SPU, which are VCP specific.

minimum guaranteed BW=20%
maximum allowable BW=50%
call block threshold level=2 (65%)

In each of the four scenarios of FIG. 9, the incoming FRR is a request for additional bandwidth on a link of the VCP for the node addressing the request. In the case represented by FIG. 9(*a*), the FRR is rejected by the node because the maximum link utilization threshold is 80% (this "threshold" is the measurement of actual bit usage as a percentage of link bit rate, or current bandwidth used), and here, only 20% of the available (unallocated) BW remains on the link. Hence, the link utilization is currently at the maximum threshold (for this link), and in that situation no T-slot requests are granted if to do so would cause the minimum guaranteed bandwidth to be exceeded for the requesting T-slot type. Here, the minimum guaranteed BW for the latter is 20%, and the percentage of BW allocated to channels of the latter is 25%. Therefore, the request is rejected (denied), even though the current allocated BW is less than the maximum allowable BW for the requesting T-slot type.

In the situation represented by FIG. 9(*b*), the FRR is rejected because the requesting T-slot type has already reached its maximum allowable BW (50%). In FIG. 9(*c*), the FRR is rejected because the current BW usage on the link is at the call block threshold level (measurement of total BW in use by all T-slot types on the link, here 65%, and which is the same as the link utilization threshold level), and the requesting T-slot type has already exceeded its minimum guaranteed BW.

In the scenario of FIG. 9(*d*), the FRR is granted by the node. Here, even though the link is at its maximum utilization threshold (80%), the requesting T-slot type is below its minimum guaranteed BW (10% versus 20%). Since the other 70% of BW up to the maximum utilization threshold is currently allocated to channels of other T-slot types, bandwidth seizing (to be discussed in greater detail presently herein) will be triggered.

The bandwidth allocation in response to FRRs has an impact on other T-slot types. That is, when channel requests for a "dormant" T-slot type begin increasing (up to its minimum guaranteed BW), the total link activity level may push other T-slot types, which are highly active, over their call block threshold. In that case, call blocking will be initiated on those other T-slot types, even though their respective activity levels on the link is not increasing. Call blocking will continue until the reserved BW of the other T-slot types falls to their respective minimum guaranteed BW levels. Call blocking (and BW seizing) will, however, take T-slot type priority level (gleaned from the PFC field) into account, so that the T-slot types of lower priority are "bumped" from their excess BW before the T-slot types of higher priority.

Table VI below illustrates subscriber traffic throttling techniques for composite data frame bandwidth management on trunk lines (links), and the relationship between call blocking and other traffic throttling techniques.

TABLE VI

Throttling	Scope	Impact
1) call blocking	per T-slot basis - at T-slot specific link utilization level (prioritized)	no new connections granted quality of service to existing connections not affected
2) BW seizing	per T-slot basis - at max link utilization level only (prioritized)	call blocking quality of service to existing calls is degraded no data dropped by network
3) frame discard	indiscriminant (all T-slot types) occurs during link transmit queue congestion	call blocking, all T-slots data is discarded without consideration of T-slot type or priority

The bandwidth seizing process is implemented according to the following algorithm. A given T-slot type channel will be allocated on a VCP at the expense of seizing bandwidth from other T-slot types if:

(1) BW reserved for the requesting T-slot type<minimum guaranteed BW for this T-slot type; and

(2) link utilization level is at or above the maximum threshold.

The NMS transmission link specific parameters may be configured so that when two or more T-slot types exceed their minimum guaranteed bandwidth, one can continue requesting more bandwidth at the expense of triggering call blocking on the other T-slot type(s). This is achievable by setting that T-slot's threshold and maximum allowable BW to higher values. However, an FRR by one T-slot type will not be granted at the expense of bandwidth seizing unless the requesting T-slot type is below its minimum guaranteed BW level. This rule is implemented in the system of the invention because bandwidth seizing results in degraded service for already established connections.

The present invention utilizes a technique of frame compression for building and formatting the composite data frames, which, together with BW seizing, represents a difference in kind from the bandwidth contention technique taught by the '886 Patent and from other schemes taught by the prior art. For example, rather than having each burst on a subscriber connection contend for one of the statically allocated channels within the packet payload, according to this aspect of the invention any unused bandwidth is compressed completely out of the frame being launched. If a subscriber connection has data to send, it can fill its respective T-slot channel(s) on the next outgoing composite data frame being built for launching on the VCP. However, if the subscriber connection has no data to send at the time the frame is being composed, the local anchor EFPS simply compresses the frame by completely removing that channel.

This frame compression scheme unlocks bandwidth consumed by empty or partially empty payloads inherent in fixed size payload protocols such as ATM. The ISN network of the present invention utilizes a hybrid protocol in that the composite data frames are of variable rather than fixed length, but the channels within the frame are of fixed size on a per T-slot type basis. Locked bandwidth cannot be used for transporting subscriber data, nor is it even available to the subscriber data stream. It consists of (i) bandwidth consumed by framing overhead (header and trailer), (ii) a partially empty frame payload, (iii) transmission link control signalling, and (iv) network control signalling. The relatively small channel sizes in the composite data frame of the invention, compared to the prior art ISDN and ATM schemes, minimize fixed size payload utilization sensitivity to traffic characteristics (e.g., burst/block size and frequency). The frame compression technique and the structure of the composite frame tend to drive the ratio of effective bandwidth utilization (EBW), i.e., the percentage of BW consumed by the subscriber data stream, to actual bandwidth utilization (ABW), i.e., the sum of the percentage of BW consumed by the subscriber data stream plus the percentage of locked bandwidth, toward the ideal ratio of EBW/ABW=1.

Frame compression eliminates two causes of partially empty payloads, viz., (1) an empty channel attributable to the subscriber having been idle during the last frame launch period, and/or (2) the T-slot type for the channel in question is undergoing flow control on this VCP. The scheme results in omission of empty channels from the composite data frames; in effect, the frame is "collapsed" before it is launched. Neither additional control signalling between nodes on the VCP nor frame reconfiguration is required for frame compression.

In performing the preferred method of frame compression, no space is allocated for a channel by the VCP anchor EFPS during frame composition unless the associated SPU has posted a cell as being ready for transmission. In this context, "cell" refers to a unit of subscriber data that the SPU places in a predefined memory location ("bucket") which is checked by the anchor EFPS during the next frame launch. That cell will be included by the EFPS in the next frame if the SPU has posted (flagged) it as being ready. The size of the cell is equal to the size of the channel for the T-slot (traffic component) type of the particular subscriber. If a partial cell is posted, the EFPS will include and launch it in the next data frame, resulting in a partially filled channel. However, depending on the traffic characteristics of the particular traffic component type, such as delay insensitivity, the SPU may elect not to post a partial cell in the expectation that it may be filled during the next frame launch period.

If the SPU has not posted a cell for a particular subscriber, the anchor EFPS notes the absence of a flag for the bucket and sets the PFC field bit associated with the channel allocated to that subscriber (the respective "C" bit) to "not present" (i.e., C=0). In an alternative embodiment and method, if a specific T-slot type were not required to have "per channel" PFC bits (i.e., C bits), so that frame compression were not performed on a per channel basis, it would be performed on the basis that either all of the channels are present or none is present, as indicated by the "per T-slot" presence bit (i.e., the "B" bit) in the PFC field. In this respect, it should be noted that for T-slot types with very small channels and/or highly active subscribers, per channel frame compression is less effective from a bandwidth or processing standpoint than with larger sized channels or where the subscriber is somewhat less active.

FIG. 10 is a simplified diagram of a VCP anchor EFPS illustrating the launching of composite data frames utilizing the frame compression method of the invention. EFPS 120 is the local VCP anchor, which built and launched composite data frame 122 in the immediately previous frame launch period. In its header, frame 122 has a PFC field which includes, for one of the frame's T-slots, an A bit=0 (indicating no flow control), the B bit=1 (indicating presence of the associated T-slot), and a sequence of C bits all of which=1 (indicating presence of each of the three channels in the payload of the frame for that T-slot). For the frame 123 being composed during the current frame launch period, however, the SPUs of EFPS 120 have posted cells for only subscribers (channels) 1 and 3 of this T-slot. Consequently, this frame is compressed without a channel 2 as indicated by C=0 in the PFC field for this T-slot.

The anchor EFPS at the remote side of the VCP receives each composite data frame, such as 123, transported on the VCP, and interprets the payload format during frame decomposition by comparing the delta change in the PFC field of the incoming frame relative to the frame format template which was received and stored during the last frame reconfiguration. In the situation represented by FIG. 10, the remote EFPS thereby recognizes the absence of channel 2 of the first T-slot in frame 123.

In the architecture of the preferred embodiment of the multimedia transmission system according to the invention, a frame is sent even if no data is being transmitted in its T-slots. This not only informs the remote side EFPS that no channels are active, but also serves the purpose of maintaining the synchronous frame launch aspect of the invention. Otherwise, it would be necessary to dispatch time stamps to indicate the particular time that each frame is launched, to keep track of individual frames in the network for use in the frame decomposition process at the remote anchor EFPS. Synchronous launching of the frames at predetermined equal intervals of time throughout the network eliminates the need to send such additional information regarding timing. Each node that receives a frame (whether TFPS or remote EFPS) recognizes that the preceding packet wa sent one millisecond (or whatever other synchronous frame launch interval is used) before the current one.

Frame compression is a core building block of the bandwidth seizing technique. In essence, the latter is a flow control technique which is sensitive to the NMS assigned T-slot priorities, and to the unique traffic characteristics (such as delay, data loss, packet jitter, etc.) of each traffic component type. As previously observed herein, bandwidth seizing is used during high link utilization periods to temporarily reallocate reserved bandwidth from a T-slot type exceeding its maximum guaranteed bandwidth to a T-slot type which is below its minimum guaranteed level and is requesting additional bandwidth, or to provide basic flow control of all traffic components. It allows maximum bandwidth sharing and allocation, as a percentage of total link capacity, without increasing the risks of call blocking and/or unacceptable degradation of service quality.

The bandwidth seizing technique of the invention requires no additional bandwidth for its initiation, in contrast, for example, to the flow control technique utilized in an X0.25 network which requires a "received—not ready" (RNR) control packet (with its additional BW overhead) to be sent to initiate flow control. Furthermore, frame reconfiguration is not required, but frame compression is automatically triggered when the SPUs reduce traffic onto the network by not posting cells during frame composition. As illustrated by the traffic throttling techniques of Table VI above, bandwidth seizing is a more "drastic" throttling technique than call blocking, but less "drastic" than frame discard.

In processing the composite data frame on a VCP, each TFPS initiates flow control when an associated link's transmit queue size crosses a predefined link utilization threshold level indicative of congestion. This situation may arise either when (i) a new channel request is made by a T-slot type which is below its minimum guaranteed BW level, or (ii) a statistical aberration occurs in which an unusually large number of already allocated channels on each VCP are simultaneously sending data (i.e., all SPUs are posting cells for each frame launch). It will be understood, of course, that although packet switching is a statistical multiplexing approach which assumes certain averages, and not peak usage, nevertheless, peaks do occur.

When flow control is called for, it is initiated on T-slots exceeding their minimum guaranteed BW, starting with T-slots of the lowest priority. For each composite data frame in the receive queue on the congested link of the affected TFPS, the TFPS sets the A bits to 1 (indicative of flow control requirement) in the PFC field corresponding to the affected T-slot types. At the EFPS(s) that receives frames with this modified PFC field, the response is to implement bandwidth seizing, in the manner illustrated by the sequence of parts (a), (b), (c) and (d) of FIG. 11.

Referring to FIG. 11(*a*), anchor EFPSs 130, 131 and 132 for three different VCPs 134, 135 and 136, respectively, build and launch composite data frames onto the respective VCPs during each frame launch period. An SPU associated with each EFPS posts cells from each subscriber data stream to be composed within channels of the respective T-slot type in the composite data frame for launching onto the respective VCP. In this example, all three of the VCPs traverse transit node (TFPS) 138. Each of the EFPSs has two subscribers (numbered 1 through 6, respectively) providing data streams of different traffic component types. The PFC bits (only A and B are shown here, for the sake of simplicity and because the individual channel bits are not used for this activity) in the headers of the data frames 141, 142 launched by EFPSs 131 and 132 respectively, indicate the presence of two T-slot types each (both B bits=1) and no flow control for either (both A bits=0). (Also for simplicity's sake, individual channel information is not shown for the two T-slots in each of those payloads).

An FRR control frame 137 is issued by EFPS 130 requesting additional bandwidth for a T-slot type which is below its minimum guaranteed BW level at the congested link 139, which carries all three VCPs at the other side of TFPS 138. Link 139 has reached the maximum utilization threshold level on the transmit queue from the TFPS. Also, the T-slot types in the incoming frames 141, 142 from EFPSs 131 and 132 are currently exceeding their respective minimum guaranteed BWs. Under these conditions, bandwidth seizing from those T-slot types is required. According to this aspect of the invention, those T-slot types which exceed their minimum guaranteed bandwidth, and which are at a lower transmission priority level (assigned priority ranking, as discussed above) than the traffic component which is requesting more bandwidth, will have bandwidth seized from them.

In FIG. 11(*b*), the VCP anchor EFPSs are notified of the need for bandwidth seizing. TFPS 138 (and any other transit node having a congested link—maximum link utilization—and experiencing the same conditions) responds to the conditions of FIG. 11(*a*) by setting the A (flow control) bit in the PFC field(s) of the affected T-slot(s) (i.e., those of lower priority and from which BW is to be seized) to 1, in frames (e.g., 145, 146) in the receive queue at that TFPS destined toward the VCP anchor EFPSs which are generating the excessive traffic (EFPSs 131, 132 in this example), thereby requesting flow control. Only the PFC flow control bit associated with the first T-slot in each of frames 145 and 146 has been so modified, in this example. All composite data frames in the receive queue for this TFPS are affected, without regard to which of the three VCPs they are associated with. These are data frames which were launched from the remote side anchor EFPS of the respective VCP. EFPS 130, which issued the pending reconfiguration request (FRR control frame 137) will continue to launch composite data frames with the previous format for the entire period that the FRR is pending.

In FIG. 11(*c*), during decomposition of the received frames at EFPSs 131, 132, the flow control indication for the affected T-slots (one on each VCP in this example) is detected from the associated A bits in the frame headers. As a result, the fast packet internal protocol (FIP) subsystems in these EFPSs dispatch flow control command cells (indicated at 147, 148) to their respective SPUs to perform less frequent posting of cells for building and launching the composite data frames by their respective EFPSs. During this time, the congestion has not yet been alleviated on the transmit queue at TFPS 138 for link 139.

In FIG. 11(*d*), frame compression is implemented to free up bandwidth (by seizing it) from the T-slots affected by the flow control request. The less frequent posting of cells by the affected SPUs results in frame compression by eliminating some or all channels of the affected T-slot type(s) in some of the composite data frames. To that end, the algorithm utilized for the purpose may be customized on a per T-slot basis—for example, to use voice clipping techniques for discarding portions of the data stream of a voice traffic component subscriber, and to permit sending an RNR control packet by an X0.25 data traffic component subscriber because of its lower tolerance of data loss but greater tolerance of delay than the voice traffic component. It is not necessary that the transit node be cognizant of the method used at the EFPSs to relinquish bandwidth as part of the BW seizing process, and consequently, the ISN network design is simplified. That is, for example, a new traffic component type may be introduced into the network without requiring redesign of the internal/-transit network.

The freed bandwidth or a portion thereof is thereby reallocated (i.e., redistributed), and the FRR 137 which has been held by TFPS 138 is dispatched to the next transit node along the VCP(s) once the traffic profile indicates that TFPS 138 (or more accurately, its associated link of interest) is no longer congested. Composite data frames 151 and 152 launched by EFPSs 131 and 132, respectively, have flow control bits set and T-slot non-present bits in the portion of the PFC field associated with the first T-slot of the data frame, attributable to the flow control/frame compression. Although link 139 may remain at the maximum congestion threshold for a period after BW seizing is implemented, the T-slot type distribution will have changed as a result of the flow control and redistribution of bandwidth to relieve the congestion in relatively short order.

A VCP template is stored at each TFPS (and at the EFPS anchors) to describe the composite data frame format of each VCP that traverses the link(s) on which this transit node is located. The TFPS also stores the above-described T-slot type profile table, which specifies, among other things, the priority level of each traffic component type supported by the ISN network. This information allows the transit node to readily determine whether a traffic component type requesting additional bandwidth is of higher or lower priority than other T-slot types on the VCPs traversing the associated link, their respective minimum guaranteed and maximum allowable bandwidths, the state of congestion and maximum utilization of the link, and accordingly, whether or not bandwidth seizing should be invoked for flow control of one or more of the traffic components. If needed, bandwidth seizing is performed simply by changing a bit value in the PFC field to adjust the traffic flow of selected T-slot types.

Although bandwidth seizing is initiated by the transit node at the point of traffic congestion, the reduction in bandwidth usage is handled at the VCP anchor points by the T-slot-specific SPU subsystems for as long as the transit node continues to require the bandwidth seizing. Bandwidth reduction is the same percentage of maximum or targeted bandwidth for each of the respective traffic component types. For example, a 64 kb PCM voice channel which is reduced, because of BW seizing, by 50% will produce a maximum reduction of 32 kb/sec. A suitable algorithm for that purpose would change the posting of a cell by the respective SPU from an 8-byte channel every 1 ms to an 8-byte channel every 2 ms, using ADPCM techniques to compress the voice.

If the transit node determines that bandwidth on a given link is to be seized from one or more other traffic components types, the TFPS periodically examines a data frame in its receive queue for T-slots being flow controlled, and sets the PFC flow control bit to maintain BW seizing. When the link profile template indicates that BW seizing is no longer needed, this periodic examination of packets is ceased. The SPUs may be commanded to continue flow control at their respective EFPSs (by the process of less frequent posting of cells for the building of the composite data frames) until no "refresh" (i.e., no PFC bit requiring flow control) is encountered in decomposition of a received frame for a predefined time period N.

The bandwidth seizing scheme of the invention is especially effective when used in conjunction with a networkwide synchronous frame launch time. A one millisecond launch interval, for example, provides excellent bandwidth utilization and short packetization delay. Because bandwidth seizing granularity measured in seconds (typically, 15 to 30 seconds, versus a 1 ms frame launch interval) is four orders of magnitude slower than frames launched on each VCP, the transit node need not be precise about setting the PFC field and yet will still assure total control of bandwidth seizing. Indeed, the transit node may be somewhat imprecise and inconsistent with setting PFC bits in one or more frames associated with each VCP traversing the affected link within a time period N, without adversely affecting such control. For, example, assuming a 1 ms frame launch interval, and that the SPU exercises flow control for 10 seconds after the last bit requirement for same is received, an algorithm which provides that at least one packet in 10,000 packets per VCP has bandwidth seizing set in the PFC field will assure maintenance of bandwidth seizing on the link. Thus, despite imprecision, the technique provides reliable results and rapid reaction/initiation. With an average nodal delay of one ms and an average hop count (i.e., the number of nodes traversed by a VCP) of six per VCP, and a one ms launch interval, bandwidth can be seized and reallocated in less than six milliseconds. This excludes the propagation delay of the particular network transmission medium, which, for example, is the speed of light for a fiber optic network.

This form of communication via the PFC field is not a link protocol, but a network layer protocol. Any TFPS or remote EFPS may use it to notify the source anchor EFPS of bandwidth seizing. Many variations may be employed of the amount of bandwidth to be relinquished. A simple and straight-forward approach is to employ a fixed percentage reduction of traffic at each interface—for example, 25%. Another approach is to progressively reduce the traffic when bandwidth seizing is initiated and while it is in effect, and then progressively allow more traffic when BW seizing is ceased. Still another approach is based on the frequency, i.e., the percentage of data packets, in which the PFC field flow control (here, BW seizing) indication is set. Again, the communication and maintenance of bandwidth seizing requires only a small percentage of packets (one in every 10,000 to 30,000 packets would suffice) to carry the indication.

The TLS associated with the anchoring EFPS is adapted to store a VCP profile table for each VCP anchored on that EFPS. FIG. 12 illustrates the manner in which the VCP may be anchored in the EFPS. The VCP profile table to be stored preferably includes the following information: (1) the address of the remote side EFPS which anchors this VCP; (2) the location of the local VCP anchor (which may be either at a TLS or an SLS); (3) the number of subscriber VCs and their respective bandwidth requirements on the VCP (in composite data frame environment, number of channels and T-slot type); and (4) the number of other VCPs anchored on this EFPS which are also anchored on the same remote EFPS (note that two EFPSs may have more than one VCP connecting them simultaneously, because of heavy traffic between those two points). For each of the latter VCPs, information is also kept regarding the number of channels and the T-slot type of the channels.

In FIG. 12, VCPs A and B are routed through TLS 155. VCP A is TLS-anchored (to TLS 155), and VCP B is SLS-anchored (to SLS 158). VCPs C, D and E are routed through TLS 156. VCPs C and E are SLS-anchored (at SLSs 157 and 158, respectively), and VCP D is TLS-anchored (at TLS 156). The anchoring to the SLSs is through the switching fabric 160. As observed in the description pertaining to FIG. 2 above, the decision of whether to anchor a VCP on an SLS or a TLS is based on the traffic patterns between the source and destinations EFPSs. The general guidelines for arriving at this decision are as follows. First, a VCP anchor only multiplexes VCs that terminate on the same EFPS as the anchor. Second, all TLSs and SLSs include full fast packet internal protocol functionality, and therefore can anchor a VCP. Third, a subscriber data stream should not pass through the switch fabric more than once. (An exception is local switching, because the source and destination anchors are on the same EFPS).

The choice of whether to have multiple parallel VCPs between a local EFPS and a remote EFPS, and of where to locate the VCP anchor(s) (on a TLS or an SLS), is also driven by the opportunity to mutiplex (OTM) onto the VCP. A VCP is anchored on TLSs when there is OTM subscriber data streams (VCs) originating on different SLSs (such as for VCP A and VCP D in FIG. 12, where subscriber data streams from both

SLSs **157** and **158** may be multiplexed onto either VCP). This results in larger payload to header ratios, while minimizing each subscriber's packetization delay. A VCP is anchored on an SLS when there is no OTM the VCs originating on this SLS with VCs originating on another SLS, either because it is not possible or not economical to do so. It is not economical to multiplex VCs originating on different SLSs for either of the following reasons:

- payload/header ratio is so small that it is not justifiable to burden the performance oriented TLS with endpoint switching activities for this VCP;
- a VCP channel is so large (e.g., 750 kbps video) that multiplexing with other VCs is not necessary to assure low packetization delay a good payload/header ratio.

When a VCP is anchored on an SLS, the performance oriented TLS is only required to perform transit switching activities for composite data frames or control frames associated with the VCP. Endpoint processing to transit processing ratio may, for example, be in the 4:1 to 8:1 ratio.

OTM onto a VCP is a function of (i) the number of subscriber VCs going to the same remote EFPS (i.e., the number of channels), (ii) the amount of dispersement of channels over SLSs (e.g., whether all candidate VCs originate on one SLS, or on different SLSs), and (iii) the amount of bandwidth requested by the subscriber VCs (i.e. the size of T-slot type channels in the present architecture). An SLS based VCP preferably only multiplexes VCs originating on that SLS. Otherwise, as noted above, the EFPS performance is degraded because subscriber data must pass through the system switching fabric twice: once from the originating SLS to the SLS where the VCP anchor is located, and once more when the data is framed and sent to the TLS for transmission onto the trunk line toward the remote EFPS. This constitutes unacceptable overhead for the system.

A TLS anchor has the benefit of capability to multiplex onto one VCP the different subscriber connections from multiple SLSs, thus optimizing trunk line and transit node effectiveness; and the drawback of added overhead/complexity of SLS/TLS communication. An SLS anchor has the benefits that, (1) for VCPs with subscriber connections originating on the same SLS (i.e., only one SLS needs the VCP), it eliminates the overhead of TLS——SLS synchronization when the TLS has no OTM, and (2) for super-rate channels (i.e., one channel per VCP), there is again no OTM.

In the local switching example of FIG. **13**, the "VCP anchors" are at SLSs **168** and **169** of EFPS **170**, and do not require full processing. The subscriber data streams are exchanged by the SLSs using cells, without ever requiring composition or decomposition of composite data frames, keep VCP state information (templates), or many other requirements of the normal VCP processing/transmission. Clearly, this is a special case, but one worth mentioning. Both SLSs view the logical VCP as at the other side of the TLSs **171**, **172** of EFPS **170**, with both SLSs performing processing as though their subscriber data is controlled by a TLS based VCP.

The long term nature of VCPs and the inherent lack of tight coupling with current subscriber traffic (VCs) requires a periodic reevaluation of optimal VCP anchor location and VC loading (i.e., number of VCs multiplexed). As network traffic conditions change over time, the present invention allows relocation of the VCP anchor to the optimal location for those conditions. An EFPS is able to reroute VCPs, relocate VCP anchors, consolidate VCP anchors, and subdivide a VCP.

The purposes of automated VCP location will be further clarified by the following examples. As the VC load increases between an EFPS pair, multiple SLS anchored VCPs will be consolidated into a single TLS based VCP which uses the network-wide frame launch period. On the other hand, a TLS anchored VCP may be converted to an SLS anchored VCP (and even subdivided into several SLS anchored VCPs) when the VCP traffic load drops to a level that the payload/header size is unacceptably small in the network-wide frame launch period environment. An existing VCP may be rerouted/reconnected if the existing route is suboptimal because of network topology or traffic conditions at the time the route was established. It should be noted that, unlike other causes of VCP relocation, VCP rerouting may be attributable to network conditions unrelated to local subscriber traffic changes.

With respect to changing network traffic conditions which justify relocation of a VCP anchor, if it were assumed, for example, that the threshold for the function (designated θ) representing OTM onto a VCP is a constant value (designated ϕ) the conditions for VCP anchoring on the TLS or the SLS may be presented as follows:

$$\theta<\phi$$

$$\theta>\phi$$

Particular values for ϕ are to be defined for each individual implementation case. FIG. **14** illustrates a hypothetical case of the anchoring process in real time. The VCP should be anchored on the TLS as soon as θ crosses and exceeds ϕ, and should be anchored on the SLS when θ falls below ϕ. According to an aspect of the invention, either of two basic approaches may be used to trigger anchor relocation, viz.: (1) relocation on demand, and (2) periodic relocation.

In the relocation on demand approach, anchor location is reevaluated during each VC call request from a subscriber. After receiving the call request, the SLS retrieves descriptors S, R and C and makes a routing decision defining a fourth descriptor T. R is the address of the remote EFPS which anchors this VCP; S and T represent the SLS and TLS (one or the other), respectively, for the location of the local VCP anchor; and C represents the number of channels for a given T-slot type. These constitute the four descriptors of the VC/channel. At the request of the SLS, the TLS checks the anchoring conditions and reports its decision back to the SLS, namely, either (i) open a new or join the existing VCP with the TLS anchor or (ii) open a new or join the existing VCP with the SLS anchor. This decision is based on the state of the value of θ relative to the threshold level ϕ illustrated in FIG. **14**. (The activity/decision can also trigger VCP consolidation/fragmentation as shown in the flow diagrams of FIGS. **15** and **16**, described below).

In the periodic relocation approach, the relocation occurs on a fixed time interval or time of day basis. A principal purpose of this approach or mode is to correct suboptimal anchoring. Corrections must be made because the number of active channels can change over time regardless of the arrival rate of call requests. Assume, for example, that a call clear on one VC results in a value of θ which is below ϕ (FIG. 14), and that this conditions prevails for a relatively long time interval Periodic relocation corrects this situation by re-anchoring the corresponding VCP from the TLS to the SLS. The accuracy of the process depends upon a fixed time interval (or fixed time of day) for the relocation. A trade off between frequency of relocation and accuracy of the process should be made for each individual implementation/network. Unlike relocation on demand, the periodic relocation mode, in which the relative values of θ and ϕ (i.e., θ/ϕ) are periodically analyzed, reduces the possibility of thrashing between anchor locations.

The processing logic is illustrated in flow chart form in FIGS. 15 and 16. Referring to FIG. 15, which is a simplified example of adaptive anchoring showing relocation on channel request, the SLS receives a call request from a subscriber and retrieves the descriptors S, R and C, and then makes a routing decision defining the TLS number T. The SLS then sends the channel request to the chosen TLS. The TLS performs bandwidth verification, and if no bandwidth is available the channel request is denied. However, if bandwidth is available, the anchor location choice of SLS or TLS is based on the value of θ relative to ϕ. If, as a result, the SLS location is selected, the SLS is notified to build or to utilize an SLS based anchor. If the TLS location is selected, a decision is made as to whether consolidation is required. If not, normal Frame Reconfiguration is begun for replacement channels (those being consolidated). If consolidation is needed, a TLS anchored VCP is established and all SLS based anchors to be consolidated are notified. Then normal Frame Reconfiguration is begun for the original request, and a wait is instituted for other SLSs to request replacement channels (those being consolidated). Thereafter, the same logic path is followed as though no consolidation were needed.

FIG. 16 is a simplified exemplary flow chart for adaptive anchoring with relocation as a result of channel release. The SLS receives a channel release request from a subscriber, and such request is always granted in the interest of making additional bandwidth available. The TLS is notified and begins normal Frame Reconfiguration for channel release. A decision is made of whether VCP fragmentation is needed, and, if so, all SLSs involved in the VCP are notified, a wait is instituted for channel release requests from the SLSs, the requested channel is released, and if no channels are left a normal VCP tear down is begun. If, however, channels are left, the logic reverts to a wait for channel release requests from the SLSs.

Some additional considerations pertinent to BW seizing and flow control will now be discussed with reference to FIGS. 17-19. Referring to the flow chart of FIG. 17, the A bit set-up procedure is performed at the source anchor EFPS or any TFPS on the VCP. The procedure begins when an FRR control frame is received by the particular node. Upon receipt of the FRR, the node analyzes its stored associated link/T-slot profile which is indicative of traffic on the link and specific subscriber activity. If flow control is not required (A=0 in the portion of the PFC field associated with this T-slot), the T-slot profile is updated or the bandwidth reserved for the particular subscriber(s) (channel(s) is released at the node. The FRR control frame is then sent to the next node along the VCP path.

If, however, flow control is required, bandwidth seizing will be performed at the node. In that event, the A bit in the PFC field is set to 1 for all VCPs (not merely the VCP from which the FRR was received) using the particular link for all frames being transmitted (traveling) in the direction opposite from that of the FRR. Thereafter, periodic link/T-slot profile verification is performed at the node.

According to a feature of the system of the invention, a timer is utilized to determine whether or not flow control should be maintained. Referring again to the flow chart of FIG. 17, at the moment that flow control is set up, the timer is started to require periodic profile verification, meaning that the link/T-slot profile is analyzed periodically at the node. If bandwidth seizing is required, the timer continues into the next cycle. As noted above, any transit node (TFPS) along the VCP can set the A bit (to 1) and thereby turn on the flow control. Network transmission is taking place at very high speed, with a considerable number of frames moving in both directions on any given link and across any given TFPS. Therefore, it would be an undue burden to require the TFPS to determine what the state of the A bit should be in each incoming frame by performing the necessary analysis, and then set or reset flow control for each as required. The result would be thrashing at the node as it analyzes the template and issues the appropriate A bit setting for each T-slot in every frame during traffic congestion, leading to even greater congestion.

To avoid or alleviate that situation, the timer is used to provide a somewhat imprecise but nevertheless simple and effective technique by which to maintain the appropriate state of flow control at the node. This aspect of the invention recognizes that flow control is likely to be needed for more than simply one millisecond (or specifically, the particular interval which has been selected for the network wide synchronous frame launching), and that processing should be minimized at each node to the extent possible. During frame decomposition (in which the packet is taken apart to recover the payload information at the destination side), if flow control had been set for a particular T-slot an assumption is made that flow control is to be maintained for that particular T-slot in every incoming frame on that VCP for a period of time. This period of time is set by the timer. Then, even if no further frames requiring flow control for the T-slot in question are encountered on that VCP, the flow control is nevertheless maintained throughout the timer period. If, before the timer expires, a frame is received on that VCP in which the A bit is set to 1 for that T-slot, the timer is thereupon reset to maintain flow control for another timer period.

For example, if the synchronous frame launch time is selected to dispatch a packet once every millisecond, 1000 packets will be received every second. If the flow control timer is set for a period of one-half second, then 499 packets can be processed on that VCP without a requirement that the A bit must be set on any one. This technique avoids an extremely process-intensive activity at the node, while still maintaining flow control as appropriate. If the timer expires for lack of subsequent receipt by the node, during the timer interval, of another frame in which the A bit is set to 1, then flow control is lifted. The timer technique is forgiving, in the sense that another A bit may be set to 1, if necessary, when time is available to do so at the transit node. But if traffic congestion ceases there is no need to reset a bit, because the flow control will be lifted automatically when the timer expires. Further, the technique doe not require more elaborate measures such as counting the number of packets received since the flow control was turned on.

If traffic congestion is occurring at a TFPS of a VCP, the TFPS has the capability to set the A bit to 1 for the congested T-slot in each frame transported on every VCP which traverses the associated trunk (link), and this starts the bandwidth seizing process. Simultaneous with the initiation of bandwidth seizing, the node establishes a periodic link/T-slot profile verification. Referring to the table of FIG. **18**, the link/T-slot profile is built based on three parameters, designated alpha, beta and gamma. Alpha represents the total bandwidth (BW) in use on the particular link; beta, the BW usage for the particular T-slot; and gamma, the minimum guaranteed BW for this T-slot type. Alpha is 0 if the total BW in use is less than the maximum utilization threshold; beta is 0 if the T-slot BW usage is less than the maximum allowable usage; and gamma is 0 if the T-slot BW usage is less than the minimum guaranteed BW for this type of T-slot. If the reverse is true for a parameter, then the value of that parameter is 1. The table illustrates the value of the flow control (A) bit for each of several distinct and different situations represented by the value of those three parameters. The situations numbered 2 and 6 in the table are not valid, i.e., cannot exist, and therefore are disregarded.

In those situations requiring active flow control, represented by #s 1, 3, 4 and 5, the TFPS sets the A bit to 1. For example, if situation #1 exists at the node, in which total BW in use is more than the maximum utilization threshold, T-slot BW usage is more than the maximum allowable usage, and T-slot BW usage is more than the minimum guaranteed BW for this T-slot type, the node and its associated links are experiencing severe traffic congestion. Hence, flow control is urgently required. The situation may have arisen from the need to add a new channel (and thus a need for additional bandwidth), as indicated in an FRR received at the node, which triggers BW seizing.

An FRR is communicated in the form of a packet analogous to a call setup packet, and follows the same path as the composite data frames, but constitutes a request to change the format of those frames. The existing format is indicated in a template stored at each of the nodes along the path. Another stored template indicates the amount of change of BW which is permitted for a particular traffic component, i.e., how much of the BW the particular channel may be allocated. Each of the nodes processes the FRR packet, and, if the request for more BW is approved, allocates the additional BW as requested. As noted above, a request for less BW is always approved because of the desirability to have BW available at all times for allocation to other users, without exceeding anticipated reasonable needs in the initial establishment of the transmission facility (network).

If a traffic component A requires additional BW and at that time is not using its minimum guaranteed BW, while at the same time another or other traffic components B, C, D, etc. are exceeding theirs, or the total BW of the transmission facility has already been allocated, then BW must be seized from the other traffic component(s) lacking entitlement and allocated to traffic component A. The FRR can be approved in such circumstances, only if BW seizing can be implemented. If the attempt to de-allocate or seize BW from another traffic component is successful, the call is permitted to be set up in the sense of allocating the additional channel in the frame. Frame compression is employed to reallocate bandwidth, although frame compression may be used to advantage for other purposes as well—such as the previously mentioned subscriber inactivity.

Frame composition and decomposition will now be described based on the B and C presence bits in the PFC field. On the source EFPS anchor side, the template set-up during frame reconfiguration considers the physical capacity across the VCP for each end user connected to that VCP. During frame composition, the anchor node implements the following algorithm: For each channel and associated T-slot in the frame, if at least one channel is present set B=1, copy the "bucket" (i.e., the posted packet or cell) and set C=1 for an appropriate channel; else (if no channels are present) set B=0.

This frame composition/decomposition is shown in greater detail in the flow charts of FIGS. **19**(*a*) and **19**(*b*). In FIG. **19**(*a*), the source EFPS sets up the B and C bits depending on whether data is present for transmission, as part of the process of frame compression. At the start of frame composition, a first T-slot X is selected for analysis. If no buckets are posted for transmission in this T-slot, the presence bit B is set to equal 0 in the frame template. If, however, some buckets (say, N in number) are posted for transmission, the frame template is set up with B=1, and with C=1 for each of the N channels corresponding to the number of posted buckets. Data from the buckets is then written into the corresponding channels of the T-slot. If one of the buckets is empty, copying is unnecessary and the corresponding C bit is simply set to 0. Then (or in the case where B=0), the next T-slot (X+1) is selected for analysis and setup. If that T-slot is present, the same procedure is followed as with T-slot X; but if the T-slot is not present, the next T-slot is selected. The process continues until the last T-slot has been analyzed, and then the frame is sent. As emphasized earlier herein, assembly of the composite data frame requires that all of the data to be assembled into the frame must be destined for the same endpoint node.

It will be observed that two iterative processes are taking place during frame composition. First, for a T-slot go through each channel and copy each posted bucket in a corresponding channel, setting the appropriate C bit for each channel to 1 (or 0, as the case may be). Then go to the next T-slot, and if there is at least one channel present set the B bit to 1. The logic is the same regardless of the specific implementation.

The meaning of the "bucket" (or mailbox) may be further clarified as follows. At each VCP anchor (endpoint) node of the network a subscriber exists at one side of the switching fabric, and a trunk exists at the other side. Typically, in the system of the invention, the frame composition is performed on the trunk side as the traffic components cross the switching fabric. Every millisecond (or whatever other synchronous frame launch period may be used), the subscriber (through the SPU) posts a fixed size unit of data to be shipped across the network to a specified (addressed) destination. This procedure is performed at connection set-up time, in memory on the trunk side. During the next frame composition, the T-slot/channel template stored in memory is analyzed for each T-slot and the channels in that T-slot. For each of those channels a determination is then made of whether anything is posted for assembly into the composite data frame, by examining a portion of the memory in which the subscriber is to dump the data segments to be shipped. If something is to be sent, the SPU puts it into the preassigned location in memory and a flag is lifted, analogous to a mailbox flag, to indicate the presence of data to be shipped. In the preceding description, that location (and its contents) is termed a "bucket", and the bucket scheme is simply a mailbox scheme used by the subscriber to post the data which is to go into the next composite data frame.

The frame compression technique of the invention is independent of the traffic component type. If the subscriber has something to send, regardless of the traffic component type that "something" may be, a channel will be made available; and if there is nothing to send, the frame will be compressed accordingly, because there is no transmission of a blank channel.

Referring to FIG. 19(*b*), at the destination EFPS the presence bits are analyzed for purposes of the frame decomposition process. When a frame is received, the node initially examines the PFC field for the value of the B and C bits to interpret the payload structure for that incoming frame. The frame template stored at the node is used for offsetting the starting bit for each T-slot, and for each channel present in a T-slot in the payload of the frame. The frame payload is then decomposed by separating and forwarding each channel to the appropriate subscriber on the remote (destination) side. The separated channels traverse the switching fabric and travel to their respective destinations, which are mailboxes on the remote side of the subscriber connection.

The A bit is not examined during either decomposition or composition of the composite data frame. Only the B and C presence bits of the PFC field are used for the decomposition process. As has been described herein, the A bit stores information significant for a different process, that of flow control at the transit nodes, which will subsequently affect the composition process at the local EFPS side of the VCP anchor. At most, during decomposition of a frame the A bit may be observed and used to notify the subscribers that flow control is being exercised on their data. It then becomes the subscriber's responsibility to send less data.

FIG. 20 illustrates the flow in retrieval and delivery of data by the destination EFPS, using the stored VCP template. Channels A, B and C are present for one T-slot, channel D for another T-slot, and channels F and G for the remaining T-slot of this exemplary frame. No channel E data is present, hence C=0 for that channel, and the remote EFPS recognizes that channel does not exist in the payload of the incoming frame. The header address and payload information for the several channels is then dispatched to the switch fabric where the information is directed to the appropriate subscribers.

Although a presently preferred embodiment and method of the invention have been disclosed herein, it will be apparent from a consideration of the foregoing disclosure by those skilled in the field to which the invention pertains, that variations and modifications of the described embodiment and method may be made without departing from the true spirit and scope of the invention. Accordingly, it is intended that the invention shall not be limited except as required by the appended claims and the rules and principles of the applicable law.

What is claimed is:

1. A method of transmitting information between a multiplicity of subscribers, as components of traffic in an integrated services network (ISN), in which the information traffic consists of a multiplicity of media types according to the communications services required by the different subscribers, including voice, video and data traffic component types, said method comprising:

assembling a plurality of traffic component types in information streams from subscribers associated with an entry node of said ISN into composite frames of variable size for sequential launching of the assembled composite frames into the ISN destined to subscribers associated with another node of the ISN,

limiting the traffic component types assembled into each of the composite frames to those in information streams destined for subscribers associated with the same exit node of the ISN, at which the composite frames are to be disassembled,

configuring each composite frame with the traffic component types assigned to respective separate groups of adjacent channels of bandwidth allocated according to predetermined communication requirements of the particular traffic component type through the ISN, with each group limited to channels transporting traffic components of the same type, and each channel in a group dedicated to a particular subscriber of the respective traffic component type for a communication session, and

selectively seizing bandwidth from at least one group of channels associated with a traffic component type for reallocation to at least one other group of channels associated with a different traffic component type in the composite frames being launched into the ISN for preferential transmission of the latter traffic component type during periods of traffic congestion in the network between the entry node and the exit node.

2. The method of claim 1, further including:

allocating different minimum bandwidth availability levels to the various traffic component types within the composite frames to be assembled, and wherein

bandwidth is selectively seized from a group of channels associated with a traffic component type having one minimum bandwidth availability level, for use by a traffic component type having a relatively higher minimum bandwidth availability level, within the launched composite frames.

3. The method of claim 2, wherein

the step of bandwidth seizing is performed before launching each composite frame into the ISN.

4. The method of claim 1, wherein

the step of bandwidth seizing is selectively performed by selectively eliminating at least one channel of a group associated with one traffic component type within a composite frame during assembly thereof, to increase the bandwidth available in that composite frame for the group of channels associated with another traffic component type to accommodate preferential transmission through the ISN of the latter traffic component type having a higher priority for reduced delay in information transmission.

5. The method of claim 4, further including:

identifying in the header field of the respective composite frame each channel which has been eliminated therefrom, for purposes of disassembling the composite frame at the exit node of the ISN.

6. The method of claim 1, wherein

each of the composite frames is of variable size and the channels within each composite frame are of the same fixed size for any given traffic component type.

7. A method of multimedia information communication between subscribers associated with a pair of nodes of a fast packet switched network to provide traffic flow control in the network, said nodes being connected by an end-to-end network path having multiple transmission links, and wherein the multimedia information includes a plurality of traffic component types from among voice, video and data to be communicated between subscribers associated with said nodes, said method comprising:

launching a succession of composite frames conveying multimedia information from subscribers at one of said nodes intended for subscribers associated with the other of said nodes onto said network path, in which each of said composite frames has a plurality of channels of different fixed sizes to accommodate the traffic component types and representing differing bandwidth requirements allocated for respective ones of said traffic component types within each composite frame,

assigning each of said traffic component types a level of priority for transmission through the network which may differ from priority levels assigned to other traffic component types, before commencing the frame launching, and

reallocating bandwidth within newly launched composite frames according to the priority levels of said traffic component types to allocate additional bandwidth for advancing the transmission of the traffic component types having higher priority assignments across the network path while concomitantly reducing bandwidth and deferring transmission of the traffic component types having lower priority assignments across the network path during periods of traffic congestion on any of the transmission links of said network path.

8. The method of claim 7, wherein

the step of bandwidth reallocation is performed by selectively eliminating channels that accommodate traffic component types having lower priority assignments from the composite frames.

9. A system for transmitting information during call connections between a multiplicity of subscribers as components of traffic in an integrated services network (ISN), in which the information traffic consists of a multiplicity of media types according to the requirements of the different subscribers including voice, video and data traffic component types, comprising:

assembling means for assembling a plurality of traffic component types in separate respective subscriber information streams to be launched for transmission at an entry node of said ISN during respective call connections, into the message information payload field of each of a sequence of composite frames of variable bandwidth to occupy channels of predetermined limited bandwidth within the respective frame for transmission through the ISN, the assembling means including:

selecting means responsive to the respective information streams for limiting the assembly of traffic component types into each composite frame to those in information streams addressed to subscribers at a common exit node of the ISN, and

allocating means for configuring each composite frame so that the information streams of the different traffic component types are assembled into respective separate groups of channels of predetermined fixed bandwidth different from the fixed bandwidth of channels of groups associated with others of the traffic component types; and

bandwidth appropriation means located at a transit node on a communications path of the ISN between the entry and exit nodes for response to traffic congestion on said path to initiate selectively seizing of bandwidth from channels of one group for expanding channels of another group associated with a traffic component type of preferred priority of transmission within each composite frame to be launched from the entry node, to control traffic flow on said path.

10. The system of claim 9, wherein

said bandwidth appropriation means includes means for dispatching flow control data to the next node along said path to indicate the status of the flow control at the transit node where said bandwidth appropriation means is located.

11. A method of enhancing the bandwidth of certain traffic component types transmitted by subscribers as composite information in an integrated services network (ISN), in which the subscribers are associated with nodes at endpoints of the network and the composite information traffic may include voice, video and data traffic component types transmitted in packets having a predetermined variable bandwidth allocation for each traffic component type, said method comprising the steps of

detecting traffic congestion in the queue of packets awaiting transmission over a transmission link at a transit node on a network path between two endpoint nodes of the ISN,

responding to a request for more bandwidth within packets to be transmitted, for a traffic component type from a subscriber associated with one of said two endpoint nodes and communicating with a subscriber associated with the other of said two endpoint nodes, in which the traffic component type for which the additional bandwidth is requested has greater priority for uninterrupted transmission through the ISN between said two endpoint nodes than other traffic component types, according to a predetermined ranking of transmission priority levels for the traffic component types supported by the ISN,

identifying another subscriber, associated with one of said two endpoint nodes, constituting the source of a traffic component type within packets at least partly causing the congestion, of lower priority than that of the traffic component type for which more bandwidth is being requested, and

suppressing the transmission of the lower priority traffic component type from the identified subscriber in packets emanating from the endpoint node associated with the identified subscriber, to seize bandwidth from those packets, and making the seized bandwidth in the packets emanating from that endpoint node available to the traffic component type of the subscriber for which more bandwidth is being requested.

12. The method of claim 11, wherein

said step of detecting traffic congestion includes determining that the maximum available bandwidth for said transmission link is being fully utilized at the time of receipt of the request for more bandwidth.

13. The method of claim 7, wherein

the step of bandwidth reallocation is performed by detecting traffic congestion on a transmission link of the network path at a transit node between transmission links including the congested transmission link of the network path, and revising a portion of at least some of the composite frames traversing the transit node to inform one of the pair of nodes associated with the subscribers to which the revised portion composite frames are directed of the existence of the traffic congestion and the need to reallocate bandwidth to favor transmission of the traffic component types having the higher priority assignments.

14. The method of claim 13, further including

commencing a predetermined time interval during which said at least some of the composite frames are revised, without regard during such predetermined time interval to continuing congestion or relief from congestion of the transmission link that prompted the bandwidth reallocation.

15. The method of claim 14, further including

recommencing the predetermined time interval after initial expiration thereof if continuing traffic congestion is detected on the transmission link that prompted the bandwidth reallocation.

* * * * *

US005689566A

United States Patent [19]

Nguyen

[11] **Patent Number: 5,689,566**

[45] **Date of Patent: Nov. 18, 1997**

[54] **NETWORK WITH SECURE COMMUNICATIONS SESSIONS**

[76] Inventor: **Minhtam C. Nguyen**, 10018 Lexington Estates Blvd., Boca Raton, Fla. 33428

[21] Appl. No.: **547,346**

[22] Filed: **Oct. 24, 1995**

[51] **Int. Cl.**^{6} **H04K 1/00**

[52] **U.S. Cl.** **380/25**; 380/29; 380/49

[58] **Field of Search** 380/25, 23, 24, 380/4, 46, 49, 29

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,227,253	10/1980	Ehrsam et al.	375/2
5,060,263	10/1991	Bosen et al.	380/25
5,073,852	12/1991	Siegel et al.	395/700
5,111,504	5/1992	Esserman et al.	380/21
5,136,716	8/1992	Harvey et al.	395/800
5,142,622	8/1992	Owens	395/200
5,220,655	6/1993	Tsutsui	395/325
5,226,172	7/1993	Seymour et al.	395/800
5,239,648	8/1993	Nukui	395/600
5,241,599	8/1993	Bellovin et al.	380/21
5,261,070	11/1993	Ohta	395/425
5,263,165	11/1993	Jamis	395/725
5,268,962	12/1993	Abadi et al.	380/21
5,301,247	4/1994	Rasmussen et al.	380/43
5,311,593	5/1994	Carmi	380/23
5,323,146	6/1994	Glaschick	340/825.34
5,369,707	11/1994	Follendora, III	380/25
5,373,559	12/1994	Kaufman et al.	380/30
5,375,207	12/1994	Blakely et al.	395/200
5,392,357	2/1995	Bulfer et al.	380/33
5,416,842	5/1995	Aziz	380/30
5,418,854	5/1995	Kaufman et al.	380/23

OTHER PUBLICATIONS

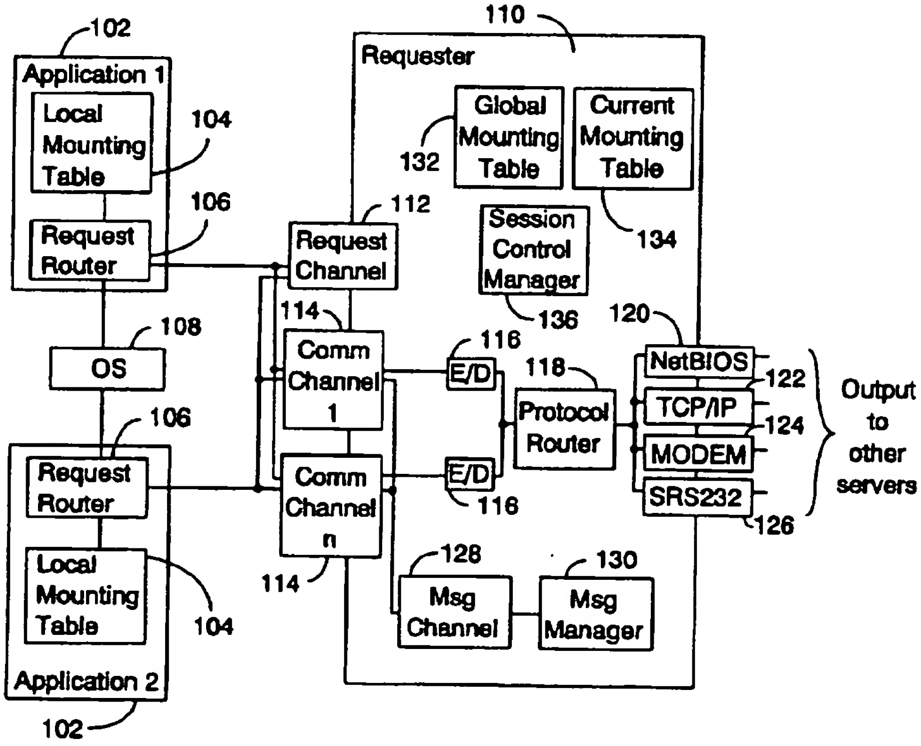
Bruce Schneier, *Applied Cryptography* (second edition), New York, NY, John Wiley & Sons, Inc. 1996, pp. 298–301.

Primary Examiner—David C. Cain
Attorney, Agent, or Firm—John C. Smith

[57] **ABSTRACT**

A system which uses three way password authentication, encrypting different portions of a logon packet with different keys based on the nature of the communications link. Nodes attached to a particular LAN can have one level of security for data transfer within the LAN while data transfers between LANs on a private network can have a second level of security and LANs connected via public networks can have a third level of security. The level of security can optionally be selected by the user. Data transfers between nodes of a network are kept in separate queues to reduce queue search times and enhance performance.

20 Claims, 13 Drawing Sheets



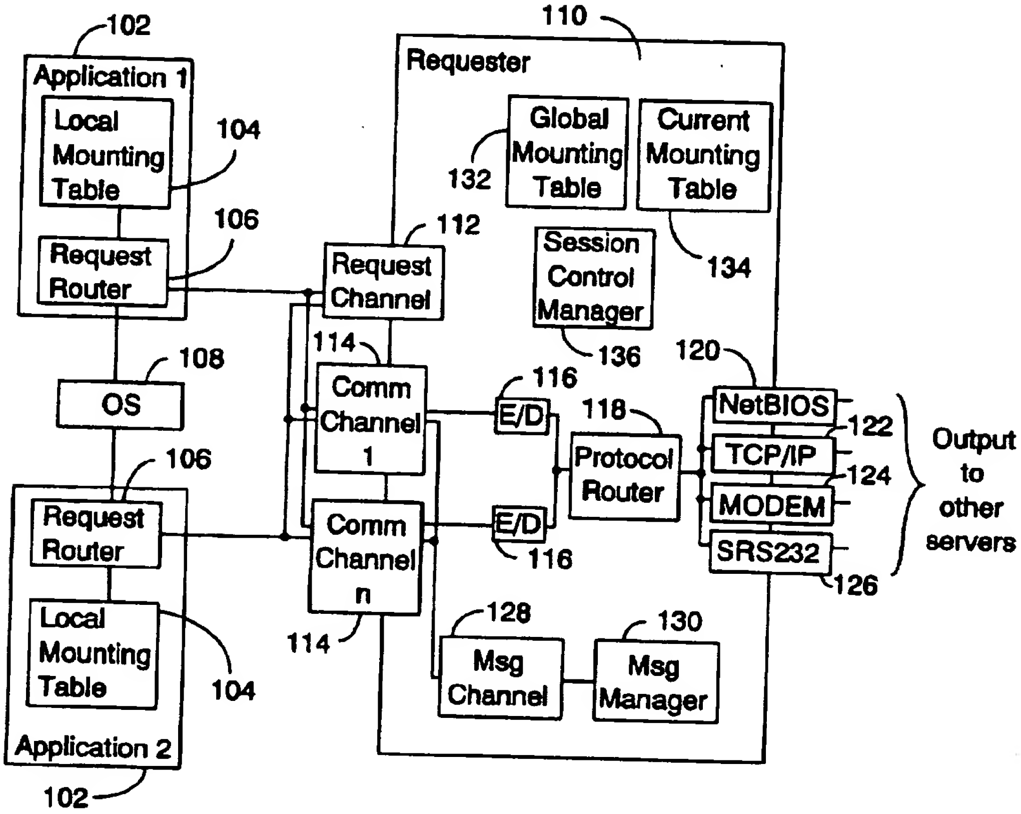
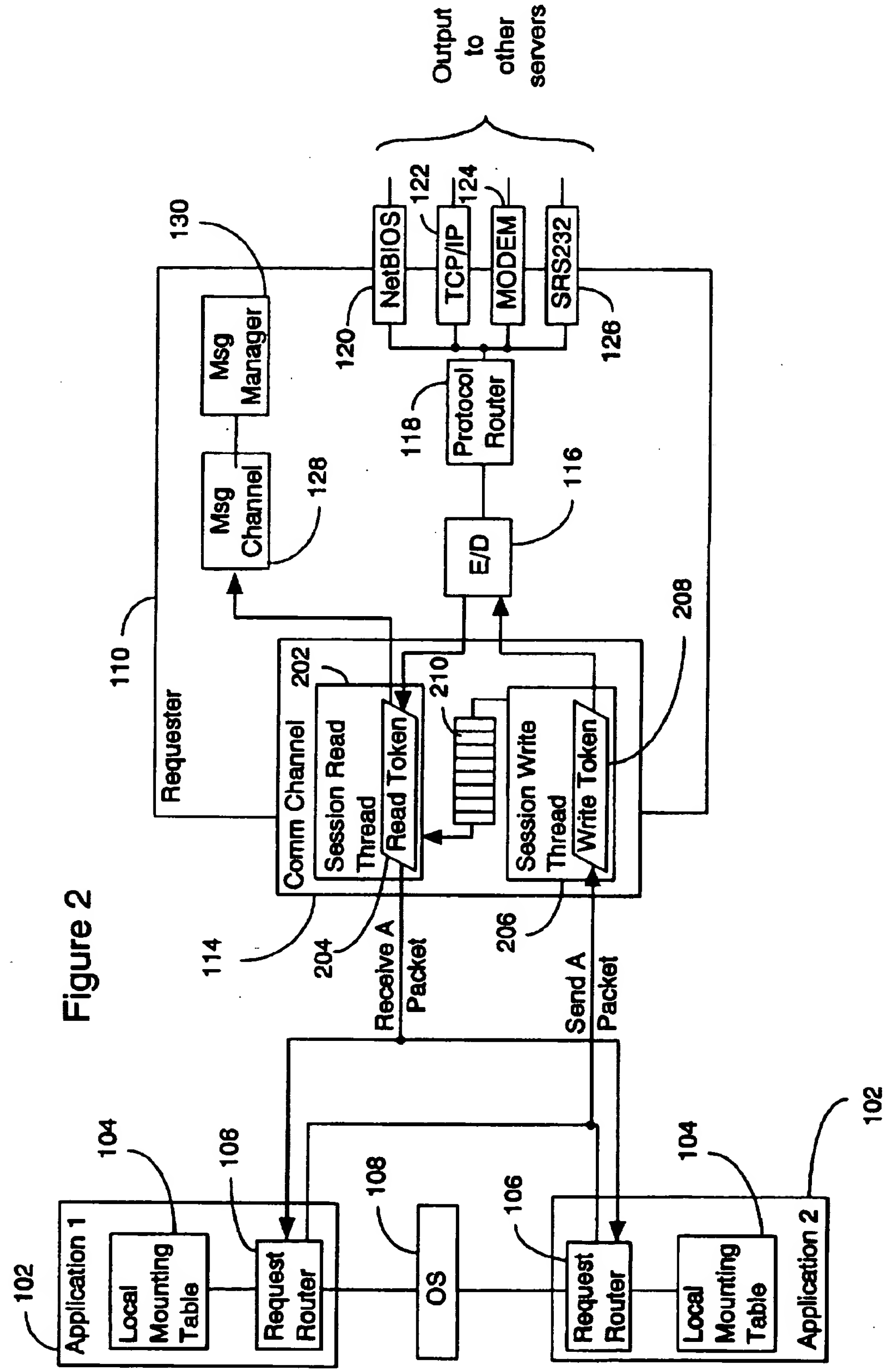

102
Application 1
Local Mounting Table
104
Request Router
106
108
OS
106
Request Router
Local Mounting Table
104
Application 2
102
110
Requester
Global Mounting Table
132
Current Mounting Table
134
112
Request Channel
Session Control Manager
136
114
Comm Channel 1
116
E/D
118
Protocol Router
120
NetBIOS
122
TCP/IP
124
MODEM
SRS232
126
Output to other servers
E/D
116
Comm Channel n
114
128
Msg Channel
130
Msg Manager

Figure 1

Figure 2


Output
to
other
servers
130
122
124
NetBIOS
TCP/IP
MODEM
SRS232
120
126
Msg
Manager
Msg
Channel
128
118
Protocol
Router
116
E/D
110
Requester
Comm Channel
202
210
208
Session Read
Thread
Read Token
Session Write
Thread
Write Token
114
204
206
Receive A
Packet
Send A
Packet
104
106
108
106
104
102
102
Application 1
Local
Mounting
Table
Request
Router
OS
Request
Router
Local
Mounting
Table
Application 2

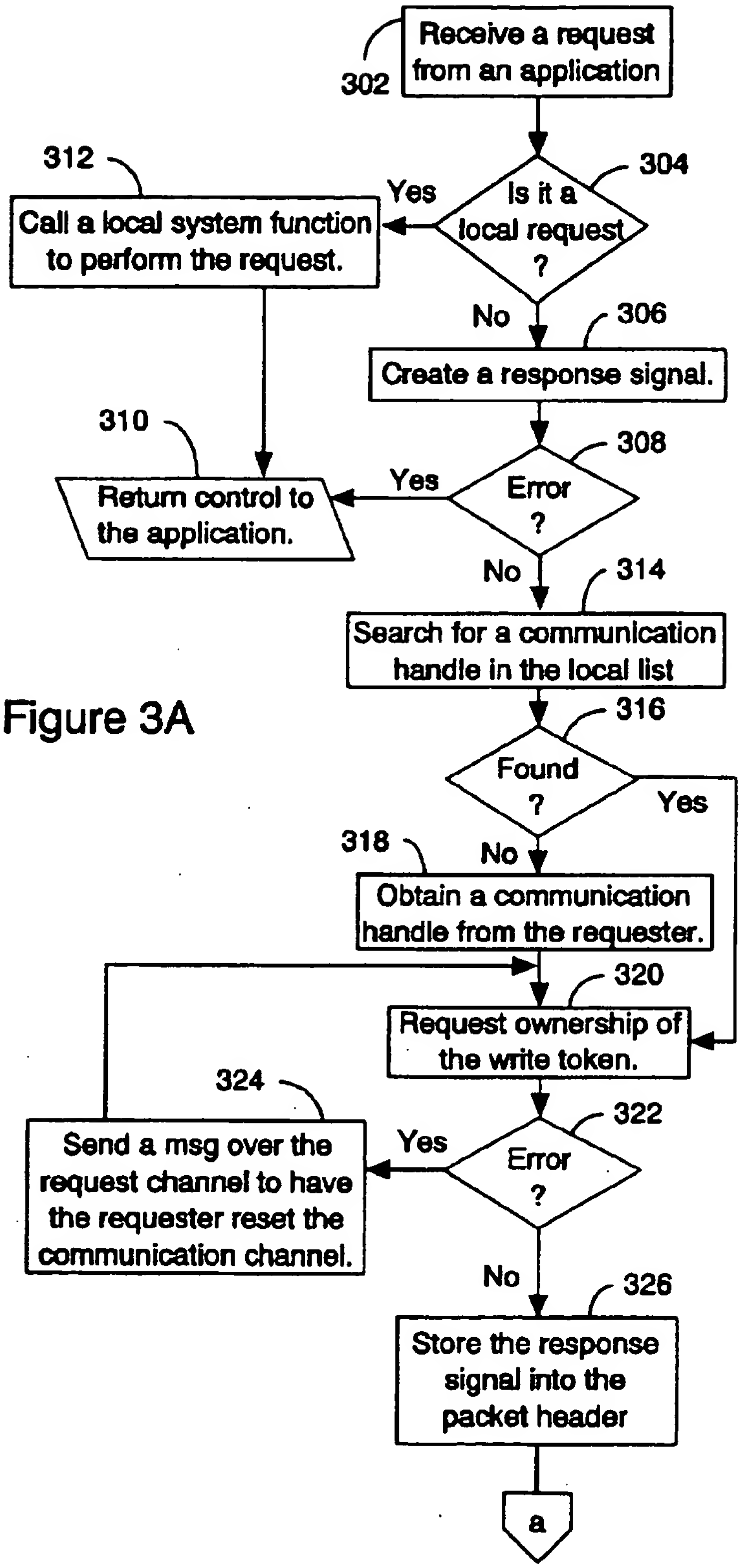
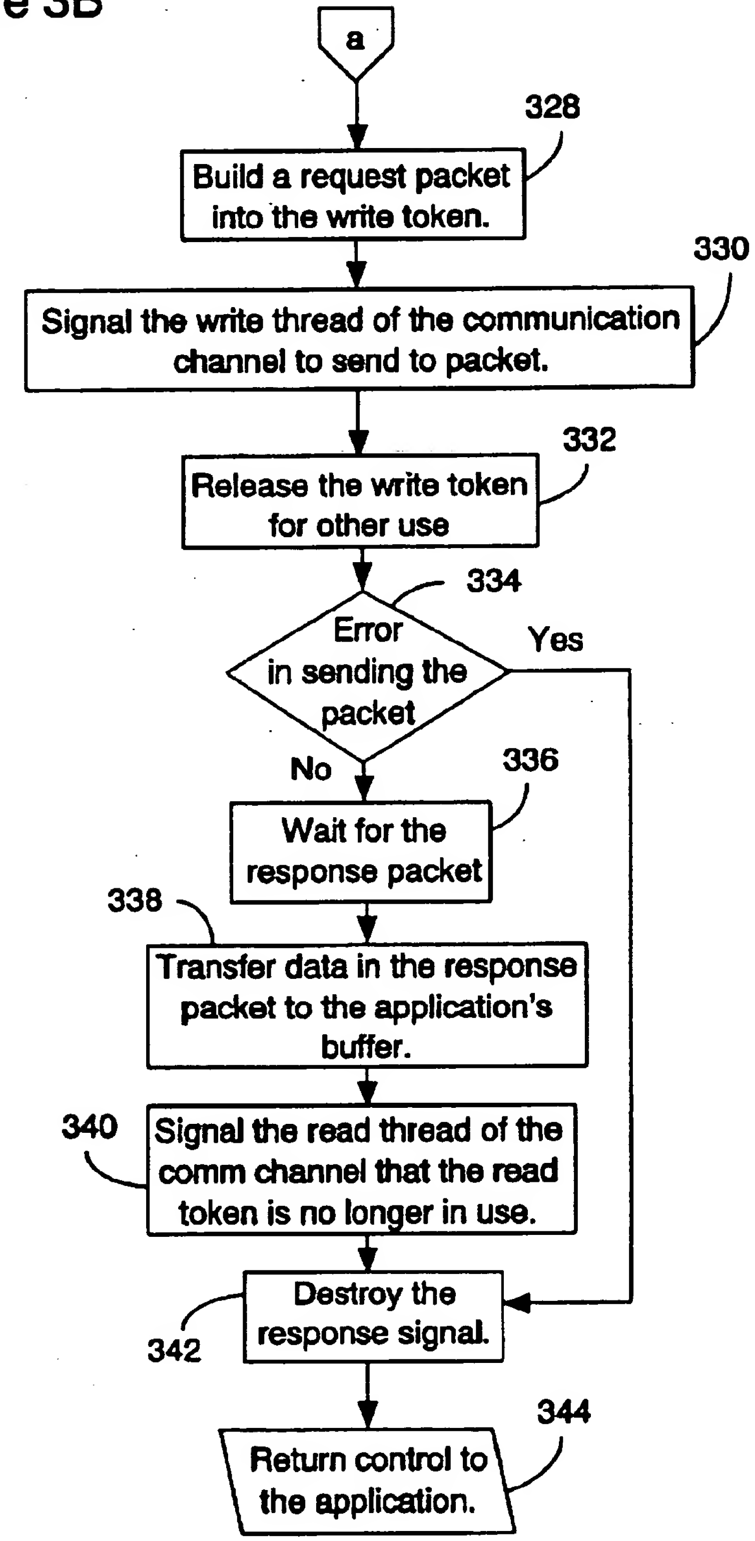

Receive a request from an application
302
Is it a local request ?
304
Yes
Call a local system function to perform the request.
312
No
Create a response signal.
306
Error ?
308
Yes
Return control to the application.
310
No
Search for a communication handle in the local list
314
Found ?
316
Yes
No
Obtain a communication handle from the requester.
318
Request ownership of the write token.
320
Error ?
322
Yes
Send a msg over the request channel to have the requester reset the communication channel.
324
No
Store the response signal into the packet header
326
a

Figure 3A

Figure 3B


a
328
Build a request packet into the write token.
330
Signal the write thread of the communication channel to send to packet.
332
Release the write token for other use
334
Error in sending the packet
Yes
No
336
Wait for the response packet
338
Transfer data in the response packet to the application's buffer.
340
Signal the read thread of the comm channel that the read token is no longer in use.
342
Destroy the response signal.
344
Return control to the application.

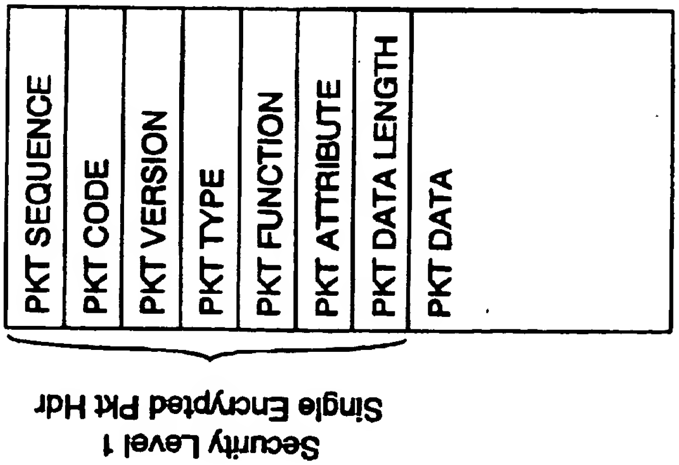


Figure 4A

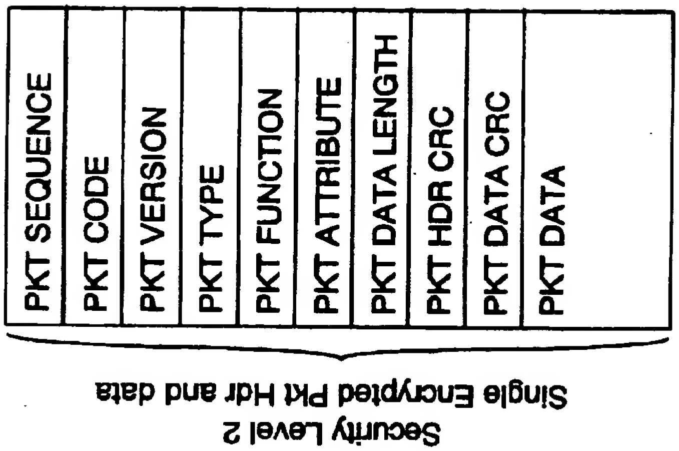


Figure 4B

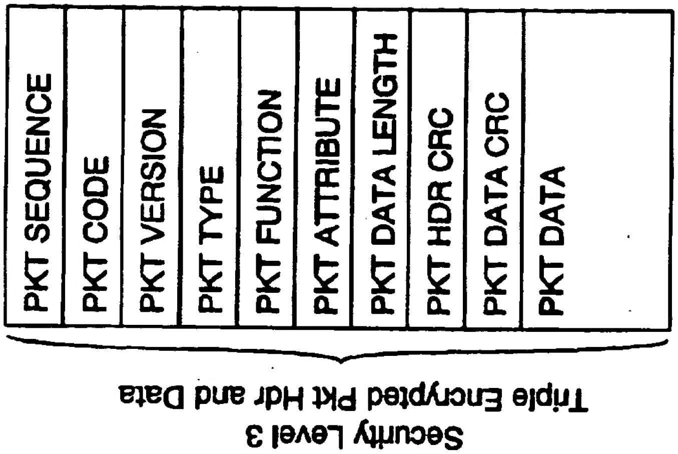
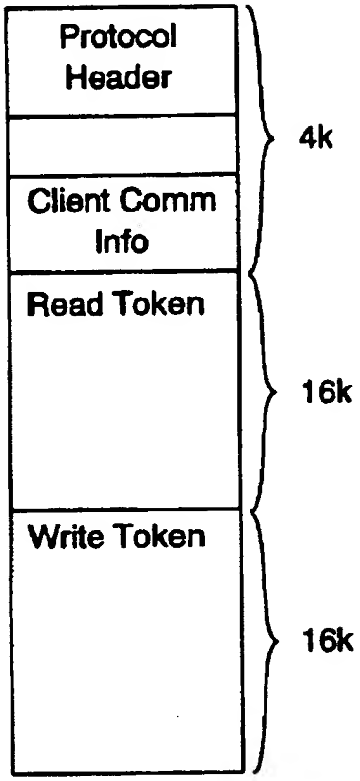


Figure 4C



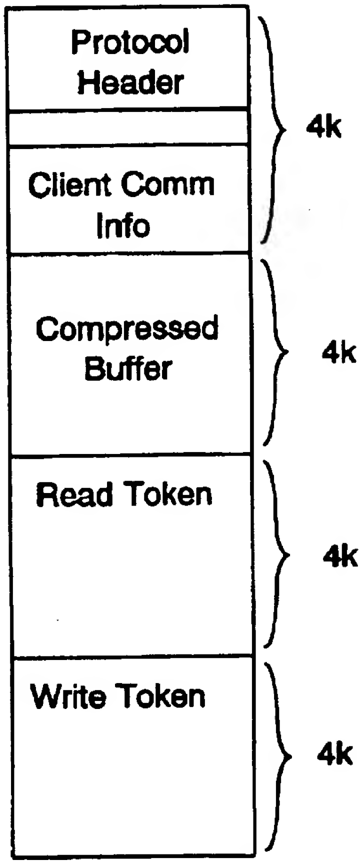


Figure 5A

Figure 5B

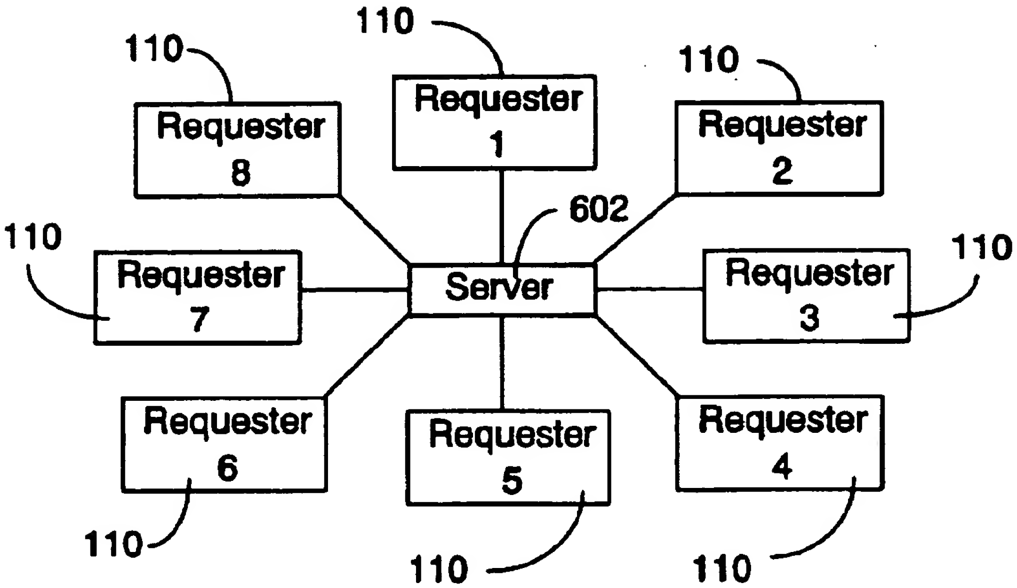


Figure 6

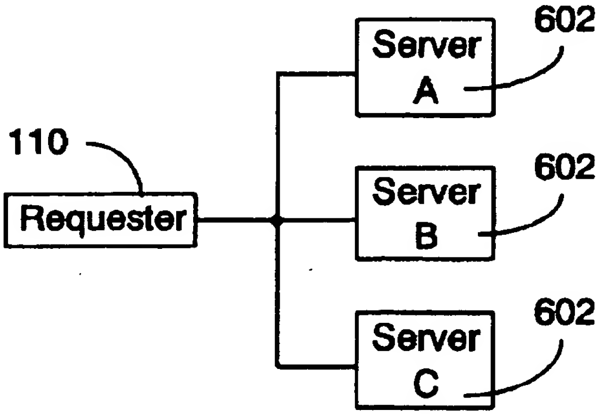


Figure 7

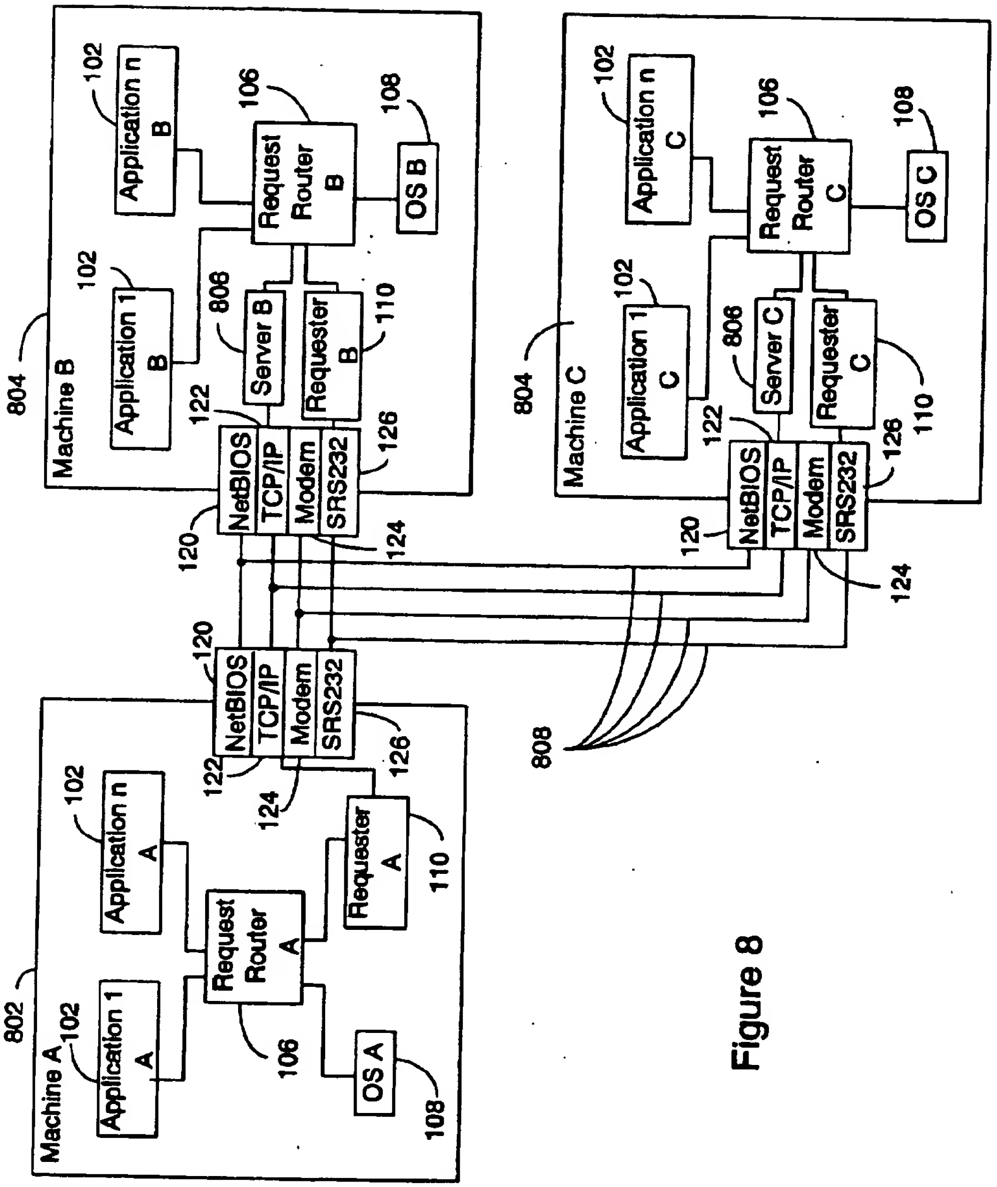

Machine B
804
Application 1
B
102
Application n
B
102
Request
Router
B
106
OS B
108
Server B
806
Requester
B
110
NetBIOS
TCP/IP
Modem
SRS232
120
122
124
126
Machine C
804
Application 1
C
102
Application n
C
102
Request
Router
C
106
OS C
108
Server C
806
Requester
C
110
NetBIOS
TCP/IP
Modem
SRS232
120
122
124
126
808
Machine A
802
Application 1
A
102
Application n
A
102
Request
Router
A
106
OS A
108
Requester
A
110
NetBIOS
TCP/IP
Modem
SRS232
120
122
124
126

Figure 8

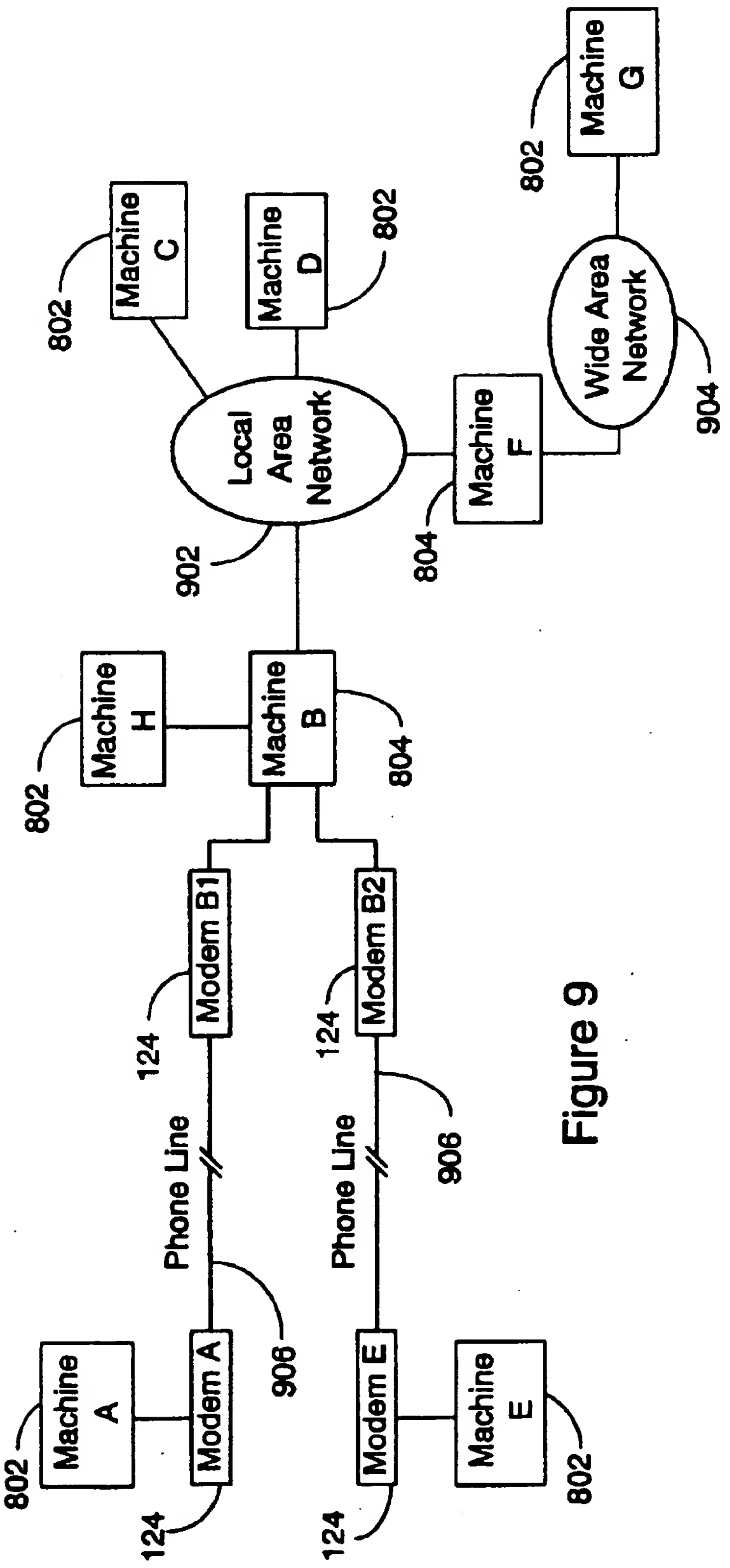

Machine G
802
Machine C
802
Machine D
802
Wide Area Network
904
Local Area Network
902
Machine F
804
Machine H
802
Machine B
804
Modem B1
124
Modem B2
124
Phone Line
Phone Line
906
Machine A
802
Modem A
124
906
Modem E
124
Machine E
802

Figure 9

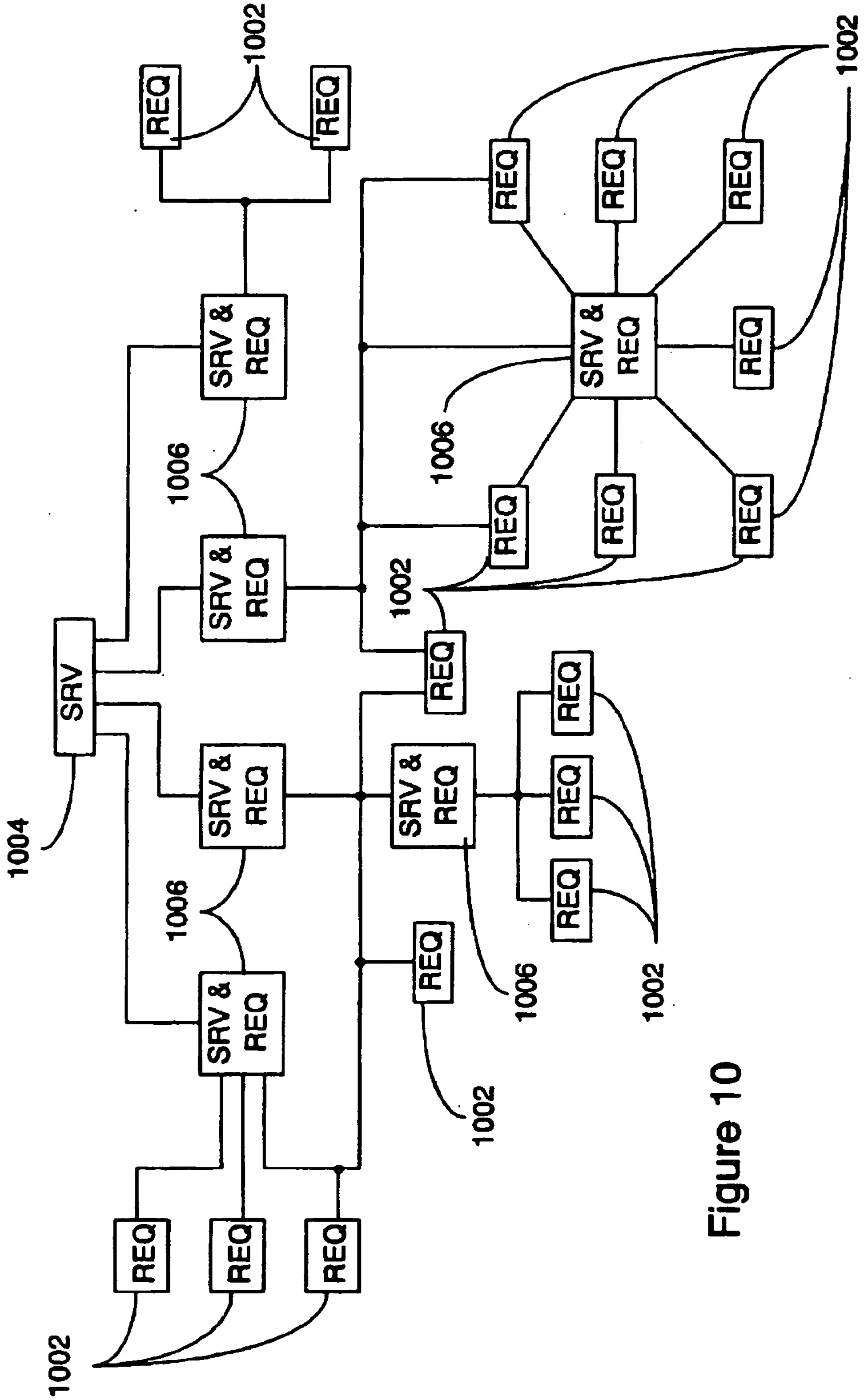


Figure 10

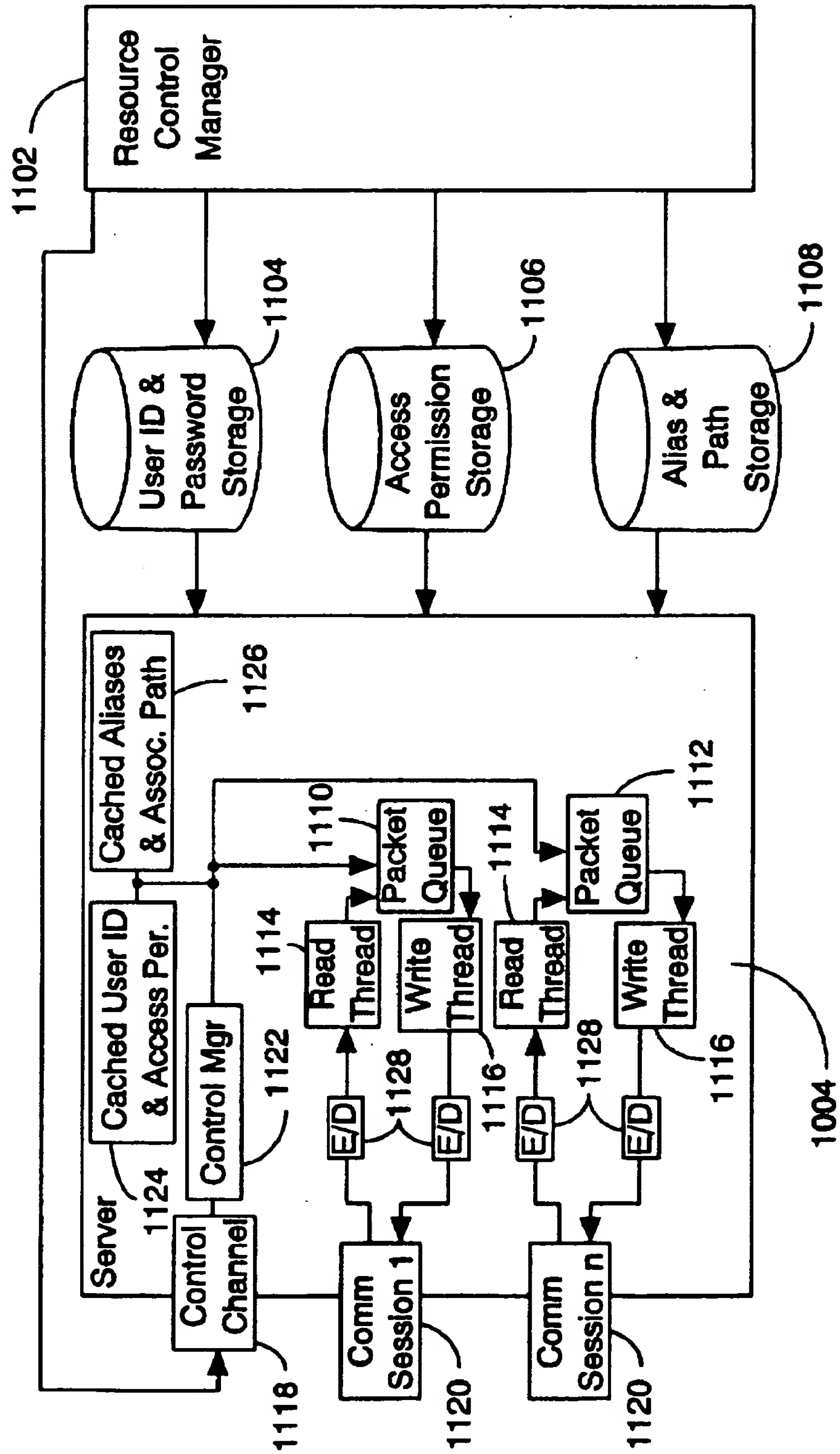

1102
Resource Control Manager
User ID & Password Storage
1104
Access Permission Storage
1106
Alias & Path Storage
1108
Server
Cached User ID & Access Per.
1124
Cached Aliases & Assoc. Path
1126
Control Mgr
1122
Control Channel
1118
Packet Queue
1110
Packet Queue
1112
Read Thread
1114
Write Thread
1116
Read Thread
1114
Write Thread
1116
E/D
E/D
1128
E/D
E/D
1128
Comm Session 1
1120
Comm Session n
1120
1004

Figure 11

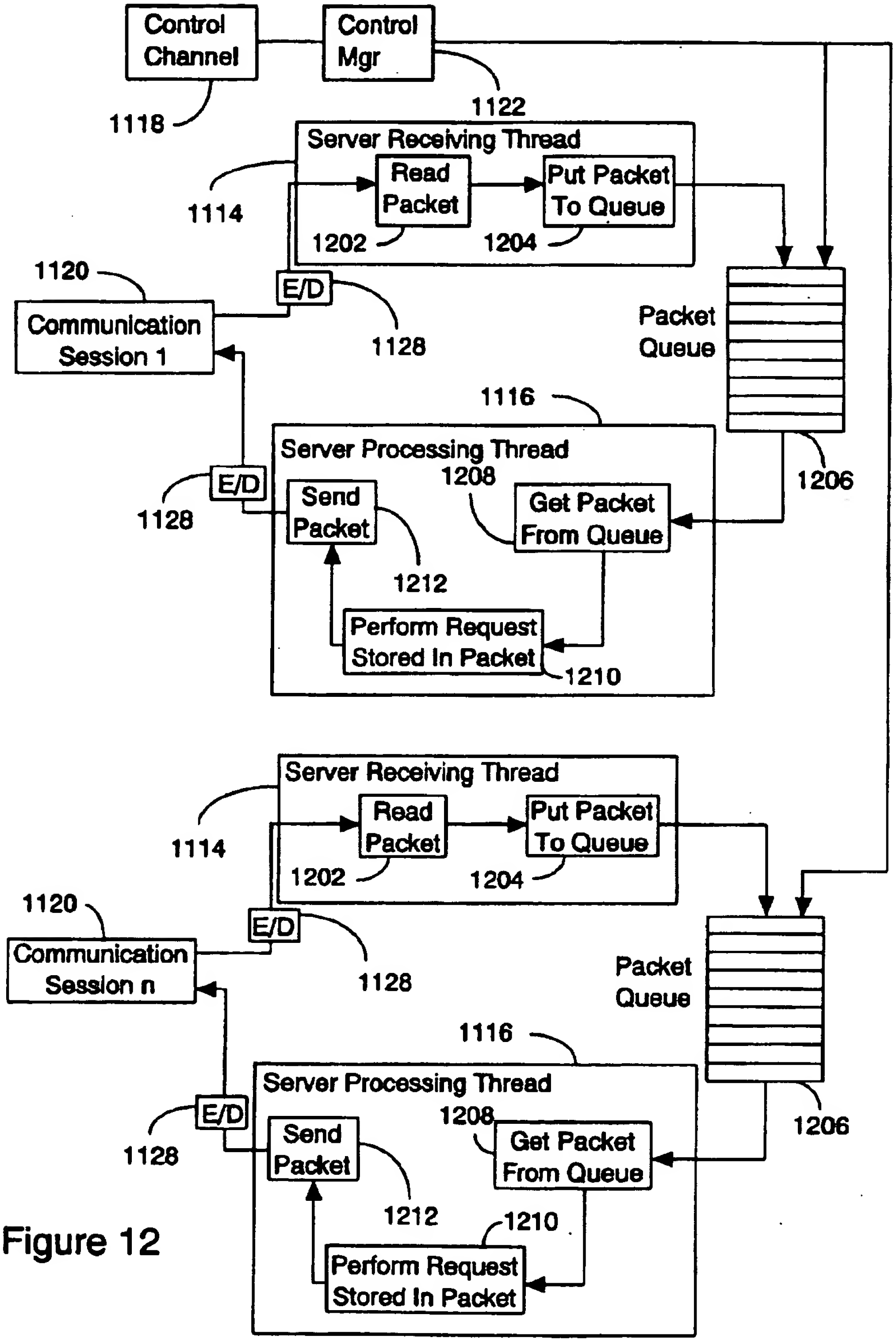

Control Channel
Control Mgr
1118
1122
Server Receiving Thread
Read Packet
Put Packet To Queue
1114
1202
1204
1120
E/D
Communication Session 1
1128
Packet Queue
1116
Server Processing Thread
1208
Send Packet
Get Packet From Queue
1206
1128
E/D
1212
Perform Request Stored In Packet
1210
Server Receiving Thread
Read Packet
Put Packet To Queue
1114
1202
1204
1120
Communication Session n
E/D
1128
Packet Queue
1116
Server Processing Thread
E/D
1128
1208
Send Packet
Get Packet From Queue
1206
1212
1210
Perform Request Stored In Packet

Figure 12

Figure 13A

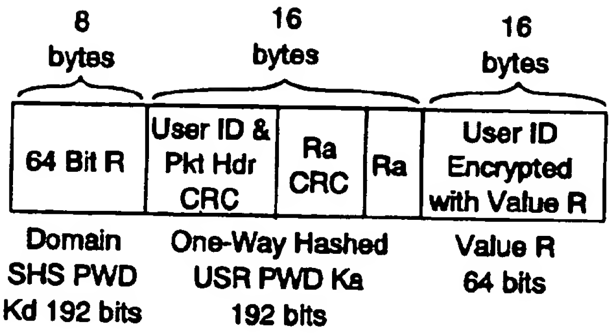
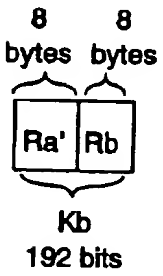

8 bytes
16 bytes
16 bytes
64 Bit R
User ID & Pkt Hdr CRC
Ra CRC
Ra
User ID Encrypted with Value R
Domain SHS PWD Kd 192 bits
One-Way Hashed USR PWD Ka 192 bits
Value R 64 bits

Figure 13B



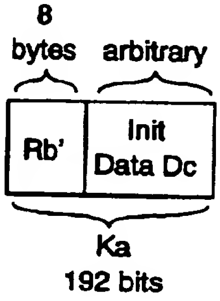


Figure 13C

Figure 13D

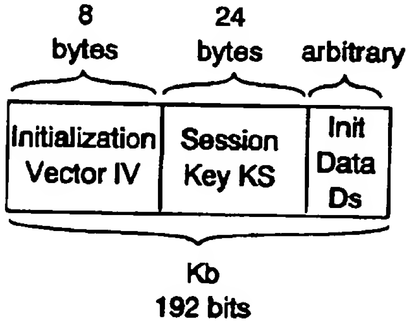
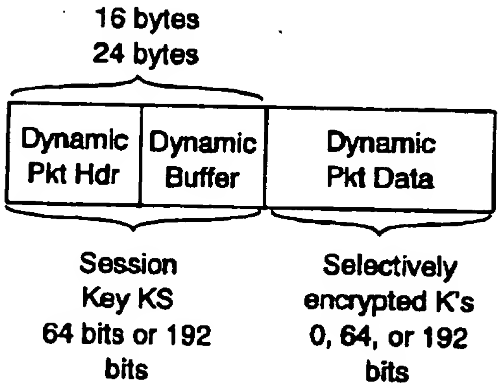


Figure 13E


16 bytes
24 bytes
Dynamic Pkt Hdr
Dynamic Buffer
Dynamic Pkt Data
Session Key KS 64 bits or 192 bits
Selectively encrypted K's 0, 64, or 192 bits

NETWORK WITH SECURE COMMUNICATIONS SESSIONS

BACKGROUND OF THE INVENTION

1. Technical Field

The present invention relates to computer network security. In particular, it relates to networks which use dynamic packet headers and multiple levels of packet encryption to transfer data to and from a remote server or to and from another node in the local network.

2. Background Art

The development of small independent systems such as personal computers has provided several benefits to users. By providing each user with their own processor and data storage, personal computers provide consistent performance and data security. A cost of these benefits is the inconvenience which results from the inability to easily access data by other members of an organization.

The use of mainframe systems, and the later development of alternative systems such as LANs (Local Area Networks) and servers reduces the inconvenience of making data available to all members of an organization, but results in unpredictable performance, and more importantly results in exposure of sensitive data to unauthorized parties. The transmission of data is commonly done via packet based systems which have user ID and password information in a header section. Interception of a packet with header information allows the intercepter to learn the user ID and password which will in turn allow future penetration of the user's system and unauthorized access to the user's data. It would be desirable to transmit user identification and password information in a manner which would be indecipherable to an unauthorized interceptor.

Data security is endangered not only by access by outside parties such as hackers, industrial spies, etc, but also to inadvertent disclosure of data to unauthorized members of the organization. For example, data exchange at certain levels of management may cause problems should the information be disclosed to the general employee population. Likewise, the transmission of personal information such as banking codes over networks has exposed individuals using online financial systems to the possibility of fraudulent access to their funds by third parties.

In addition to data security, the use of network systems such as LANs has created performance problems due to the queuing of requests from multiple locations and the unpredictable delays associated with queuing fluctuations. It would be advantageous if a system could provide not only data security, but also more consistent performance.

The prior art has failed to provide network systems which ensure that access to data is restricted to authorized parties while at the same time providing more consistent performance.

SUMMARY OF THE INVENTION

The present invention solves the foregoing problems by providing a system which uses three way password authentication, encrypting different portions of a logon packet with different keys based on the nature of the communications link. Nodes attached to a particular LAN can have one level of security for data transfer within the LAN while data transfers between LANs on a private network can have a second level of security and LANs connected via public networks can have a third level of security. The level of security can optionally be selected by the user. Data transfers between nodes of a network are kept in separate queues to reduce queue search times and enhance performance.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram showing the connection between applications and the requester in a local system.

FIG. 2 is the diagram of FIG. 1 with a more detailed view of the requester.

FIGS. 3A–B are a flow diagram illustrating data transfer between the application and requester of the preferred embodiment.

FIGS. 4A–C are diagrams of the memory layout of packet headers used in the preferred embodiment.

FIGS. 5A–B are diagrams showing the memory layout of entries in the packet queue. FIG. 5A is the memory layout used for TCP/IP and NetBIOS. FIG. 5B is the memory layout used by SMODEM or SRS232 communications systems.

FIG. 6 is a diagram of a multi-requester system with a single server.

FIG. 7 is a diagram illustrating a single requester attached to three servers.

FIG. 8 is a diagram showing a requester (machine A) interconnected with two servers (machines B–C).

FIG. 9 is a diagram illustrating multiple requesters connected to servers via local area networks (LANs) and wide area networks and public telephone networks.

FIG. 10 is a diagram illustrating multiple requesters connected to servers and server/requester systems.

FIG. 11 is a diagram illustrating the server used in the preferred embodiment.

FIG. 12 is a diagram illustrating the read/write threads and packet queues used by the server of FIG. 11.

FIGS. 13A–D are diagrams illustrating the packet headers used in the logon procedure of the preferred embodiment.

FIG. 13E are diagrams illustrating the packet headers used during data transfer in the preferred embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Prior to a detailed description of the figures, a general discussion of the operation of the preferred embodiment follows. A network can take a variety of forms. For example, it can be two personal computers communicating via modem; it can be a single LAN system within a particular facility; it can be a remote server or mainframe system with communications links to individual terminals or personal computers; it can be a network of LANs or other servers each communicating with one another or through one another; or it can be any of the foregoing systems which use not only dedicated communications lines, but also nondedicated communications (i.e. public networks such as the Internet) through a "firewall". The use of the term firewall herein refers to the requirement for increased levels of security to avoid the possibility of unauthorized data access by parties outside of the organization. Likewise, a machine in the network can act as a client or a server depending on the nature of the data transfer.

In the preferred embodiment, communication between a client and a server is as follows. The server waits for connection requests from clients on the network. The server can be started with one or more supported protocols to enable support of a variety of client types on the network.

For example, the server protocols can include, among others, NetBIOS, TCP/IP, SMODEM and SRS232. All of the foregoing protocols are well known in the art.

When a user on a client machine wishes to initiate a data transfer or other function, the client application activates a requester to access resources in the network. When the server receives a request from a client application, it activates a thread to process the request. A thread is an execution unit of an operating system. Operating systems used for this type of system are Microsoft Windows 95 (trademark of Microsoft Corporation), Microsoft Windows NT (trademark of Microsoft Corporation), IBM OS/2 (trademark of IBM Corporation). These systems may use multiple session protocols such as NetBIOS and TCP/IP or single session protocols such as SMODEM or SRS232.

In single session protocols such as SMODEM and SRS232, the same thread is used to process the request from a client since a serial port can act as a server or client, but cannot simultaneously act as a server and client. Multiple session protocols create a new thread, referred to as an original thread, and wait for a request from a client. When a request is received, the thread is referred to as a server processing thread which is used to process the client logon.

After the logon is successfully completed, the server processing thread creates a packet queue and a packet thread to receive incoming packets and place them in the packet queue. The server then waits for packets to arrive. On the client side, the client creates a session write thread to initiate contact with the server. In addition, the client creates a second thread which is referred to as the session read thread. This thread is used to receive packets sent from the server to the client.

To use resources on the network, users must first logon the server to prove their identity. A logon request is sent from the client's logon application to the requester on the client computer. Before logon data can be exchanged between the applications and the requester, a command manager is created by the requester to accept application requests. The command manager is responsible for housekeeping requests within the client computer.

In the preferred embodiment, the logon procedure uses a three way authentication to prevent the password from being transferred over the computer and also to allow both the client and the server to authenticate each other. In addition, the authentication procedure prevents unauthorized penetration of the system security by detecting the replaying of packets by third parties.

The three way authentication system encrypts the very first logon packet with different keys for each part of the packet as follows.

The first step takes place at the client computer as follows.

1—The client generates a 32 bit random number value which is concatenated to a predefined 32 bit constant to form a 64 bit value R.

2—The CRC signature C1 of the 64 bit value R and the user ID is calculated. This signature value allows detection of packet manipulation.

3—The 64 bit value R is used as a DES key to encrypt the user ID. This makes the user ID look random for each logon packet.

4—The client generates a 192 bit key K from the server name to encrypt the 64 bit value R.

5—The client generates a key Ka from the user ID and password using a one way hash function such as the Secure Hash Standard (SHS) specified in the Federal Information Processing Standards Publication 180 (FIPS PUB 180).

6—The client generates a random number Ra, calculates its CRC signature C2, and encrypts them with the signature C1 using the key Ka. This signature is used to validate the key Ka by the server.

The second step in the process takes place at the server. When the server receives the first logon packet it decrypts the packet as follows.

1—The server generates a key K2 from its machine name and the SHS to decrypt the packet header for identification. If the packet header does not contain the predefined constant, the user is unauthorized. This occurs when an unauthorized user tries to access the server over the phone line but does not know the server name (since the phone number is a public record but the server name is private).

2—If the user is authorized, the server uses the decrypted 64 bit value R in the packet header as a key to decrypt the user ID.

3—The server then uses the user ID to search a database for an access record. If the access record cannot be found, the user has entered an invalid ID and the session is terminated. If the access record is found, the server verifies if the user is allowed access to network resources at this date and time.

4—If access date and time are verified, the server retrieves an associated one way hashed password Kb from an encrypted password file to decrypt the random number Ra and the CRC signatures. The password file is encrypted with a key Kk which is selected by the system administrator at installation.

5—The random numbers Ra and the CRC signatures are then decrypted. The server calculates the CRC signature of the packet header, the user ID and the random number Ra. If the calculated signatures match the decrypted signatures C1 and C2 stored in the packet, and if password Ka matches Kb, the server manipulates the client random number Ra with a predefined formula, generates a random number Rb, and encrypts both random numbers Ra and Rb with the password Kb before sending the first logon response packet to the client.

The third step in the process takes place at the client computer as follows.

1—The client decrypts the first logon response packet.

2—The client manipulates the random number Ra with the predefined formula and compares it with the one returned from the server. If the numbers match, the client knows that it is connected to the correct server, not a fraud server from which an eavesdropper has captured transmissions from the previous logon and is echoing packets back to the client computer.

3—The client manipulates random number Rb with another predefined formula and concatenates it with the client's initiating data (i.e., the client initial packet sequence number, the encryption and compression mode for the session, and the operating system platform ID) to form a second logon packet. The operating system platform ID is useful for selecting protocols and data formats when a particular client or server is communicating with systems that may have any one of a variety of operating system software programs running. The client would typically request encryption and compression mode for the session. However, the server may indicate that the particular modes requested are not available.

4—The client then encrypts the second logon packet and sends it to the server.

The fourth step in the process takes place at the server computer as follows.

1—The server decrypts the second logon packet.

2—The server manipulates the random number Rb with the same predefined formula used by the client and verifies if the random numbers are matched. If the random numbers match, then the server knows it is communicating with an authorized client and that the first logon packet was not a replayed packet.

3—The server saves the client initiating data, generates a session key Ks and an initialization vector IV. In the preferred embodiment, Ks and IV are generated using the formula specified in Appendix C of the ANSI X9.17 standard.

4—Ks and IV are sent to the client along with the server initiating data (i.e., the server initial packet sequence number, supported and/or approved encryption and compression modes for the session, and the server operating system platform ID).

The client and server initial packet sequence numbers are used to detect packet deletion and insertion for data exchanged after the logon procedure.

The fifth step in the process takes place at the client computer as follows.

1—The second logon response packet is decrypted by the client.

2—The client encrypts Ks and IV with its own key and saves them in memory for future communication with the server. The logon procedure completes here.

After the logon procedure is successfully completed, all packet headers are encrypted using the session key Ks and the IV. The packet headers are encrypted to prevent intruders from deleting, inserting, modifying, and/or replaying the packets which may have been captured while data was exchanged over communication lines.

For ease of illustration, the following symbols can be used to illustrate the logon process:

Where:

C=a client

S=a server

E=a symmetric cryptosystem such as DES

K=an encryption key generated from the server name

R=a 32 bit random number concatenated with a predefined constant

Ka=a 192 bit key one way hashed from the user ID and password

Ra=a 64 bit random value generated by C

f( )=a hash function such as CRC to calculate the signature

g( )=a hash function such as CRC to calculate the signatures

UID=user IDs

Kb=a 192 bit one way hashed key retrieved from a database

ha( )=a hash function to manipulate the random number Ra

Rb=a 64 bit random value generated by S

hb ( )=a hash function to manipulate the random number Rb

Dc=client initial data

IV=an initial chaining vector for encryption

Ks=a session encryption key

Ds=server initial data

R'a=ha(Ra)

R'b=hb(Rb)

The logon procedure may be listed as:

1. C to S: EK(R)+EKa(Ra,f(Ra,g(R,UID))+ER(UID)
2. S to C: EKb(R'a,Rb)
3. C to S: EKa(R'b, Dc)
4. S to C: EKb(IV,Ks,Ds)

An important advantage of the authentication procedure used by the preferred embodiment is that both the client and the server verify each other as legitimate without sending the password. In addition, the use of a second set of logon packets which contain different encrypted random numbers precludes access by an unauthorized intruder who merely replays intercepted packets.

The heart of this authentication procedure is in the middle part of the logon packet, which contains the random number Ra and the CRC signatures. Since the CRC signature C2 of the random number Ra is encrypted and sent along with the logon packet, the server can authenticate the user right on the first logon packet. The manipulation of the random numbers Ra and Rb in the challenge-response fashion is to help the server defeat the replaying of the logon packet and to allow the client to authenticate the server and to defeat packet replaying as well.

The 32-bit random number in the packet header is used to make the packet header and the user ID look different for every logon packet. The one-way hashed server name is used as a key to quickly detect invalid logon packets before searching the database. This case may occur frequently when the SMODEM protocol is activated to wait for data transferred over a telephone line (i.e., a wrong number is dialed by accident or a call generated by a manual or automated telemarketing company is being received).

In addition, the server name is isolated from the user ID and password when creating a one-way hashed password to allow the portability of the database. For example, when a business grows, another server may be needed at another location and the database can be easily transferred to the new server. Of course, it would be less time-consuming to delete unauthorized users from the database than to add authorized users to the new one. To better protect the valuable information in the database, a password is required before access to the database is granted. More important, the database can be shared among servers. For example, a server Sb can receive the first logon packet and forward the user ID to a database server Sc within a private network for verification. If an access record is found and the user can access the server Sb at this date and time, the database server Sc returns the encrypted one-way hashed password Kb to the server Sb. The server Sb then continues the challenge-response as if the password Kb is returned from a local database. Note that the database server Sc encrypts the one-way hashed password Kb with the session key defined for communication between the server Sb and Sc before sending it across the private network.

In comparison to prior art systems, the design of this invention provides the server a better opportunity to resynchronize itself if the first logon packet is invalid since the receiver of the authenticating packet is in control of what is next, not the sender. On the other hand, in the prior art the sender is in control of what is next. For example, the sender generates a public key, encrypts it with a shared secret key and sends it to the receiver. If the secret key is invalid, the receiver cannot detect it. Thus, a certain number of packets must be received before the receiver can resynchronize or the receiver might have to use a timeout to resynchronize itself.

Finally, the logon protocol of the preferred embodiment is more suitable for a client/server distributed environment, because this logon protocol allows both client and server to authenticate each other without sending the user password across the communication media and prevent intruders from deleting, inserting, modifying, or replaying the logon packets. In addition, if the logon procedure fails at any point, the server releases all resources and destroys the connection without sending the response packet at that point, i.e., if the user enters a wrong server name in the very first logon packet, nothing is sent out from the server to prevent the user, a potential intruder, from knowing anything about the server. Note that this mutual authentication technique requires the client machine to have a local CPU so that the password will not be transmitted over the network before being encrypted.

The client can now perform a mounting procedure to link a network resource on the server to a virtual disk or it can identify a network resource with the following format \\servername\netname:protocol. The format allows the client to communicate with a network domain using any supported protocols. Further, this protocol can be different from the protocol used to perform the logon procedure. That is, the logon communication protocol can be different from the mounting communication protocol. Also, different virtual disks can be mounted with different protocols to different network domains. This method allows communication between a client and network domains, between a network domain and other network domains using multiple communication protocols simultaneously.

Referring to FIGS. 1 and 2, these figures illustrate the interconnection between a client and a server. FIG. 2 is a more detailed view of the system of FIG. 1.

To perform a file transfer operation, an application 102 calls a request router 106. The request router first verifies if the application 102 requests a local or remote resource. This verification is performed using a local mounting table 104 which the request router 106 obtains from the requester 110 when the application 102 is first started.

If the resource is local, the request router 106 calls a local system function call to perform the request and returns the control to the application 102. However, if the resource is remote, the request router 106 first searches its local list to see if the needed communication handle is already stored in the list. This communication handle contains information of the read 204 and write 208 tokens (shown in FIG. 2) and their associated resources. If the communication handle is not found in the local list, the request router 106 sends a message to the requester 110 over the request channel 112 to obtain the handle. Once the handle is obtained, the request router 106 creates a response signal, i.e., a return address, requests the ownership of the write token 208, stores the response signal into the packet header, builds a packet based on the application's 102 request into the write token 208, and signals the session write thread 206 of the communication channel 114 that there is a packet to send.

If the application data is larger than the packet capacity, the request router 106 can send multiple packets in a series at this point. After the packet is sent to the server, the request router releases the write token for use by another thread in the same process or a different process. If the packet was sent to the server successfully, the request router 106 waits for the corresponding response packets, i.e., a packet can cause multiple response packets returned from the server.

When a response packet arrives, the session read thread uses the response signal to tell the corresponding request router that its response packet has come and is available in the read token. At that time, the read token is accessed exclusively by the designated request router. The router then transfers data in the response packet directly to the application's buffers and signals the session read thread 202 of the communication channel 114 that the read token 206 is no longer in use so that the session read thread 202 can re-use the read token 206 for other incoming packets. Finally, after

all response packets of a request packet have arrived, the request router 106 destroys the response signal and returns control to the application 102. The final response packet is determined by a bit in the packet attribute.

The request router 106 sends a message to the command manager of the requester 110 to request the communication handle containing information of the read 204 and write 208 tokens and their associated resources. If the handle already exists, it is passed to the request router 106 immediately after the requester 110 increments the access count of the handle. However, if the handle does not exist at that time, the requester 110 will load the appropriate communication library, allocate the tokens 204, 208 and their associated resources, create a communication channel consisting of a session write thread 206 to perform auto-logon, create a session read thread 204 for the communication channel 114 if auto-logon is successful, and increment the access count of the handle before passing it to the request router 106.

After receiving the handle, the request router 106 saves the handle for use during the entire lifetime of the application. When the application 102 terminates, the request router 106 will signal the requester 110 of the event so that it can decrement the access count of the handle. When the access count is zero for a certain period of time, the session manager of the requester 110 will drop the communication session, release the tokens 204, 208 and their associated resources, and unload the communication library. Thus, this method allows resources to be allocated upon demand and released when no longer in use. Furthermore, the request router 106 can translate and format data in the application timeslices while the requester 110 is communicating with communication devices 120, 122, 124, 126 to better use the CPU time.

The request router 106 can also perform any preparation necessary to transfer the application 102 request to the requester 110 before requesting the ownership of the write token 208 to reduce the time it takes to access the write token 208. In addition, the request router 106 remembers resources for one application 102 at a time. Thus, it reduces the time to search for the needed information. With this method of sending and receiving packets, data can be exchanged asynchronously between a client and a server with minimum resources in a minimum time. In addition, request packets can be accumulated on the server for processing while the previous response packet is processed by the communication devices 120, 122, 124, 126 or traveling over the network.

Message channel 128 and message manager 130 are used to control system messages transmitted in the system. Current mounting table 134 and global mounting table 132 are used to identify usage of system resources. The session control manager is used to control each session between a client and a server.

FIG. 3A and B is a flowchart which illustrates the transfer of information in a session after the logon procedure has completed. When a resource request 302 is made, the system 304 first tests to see if it is for a local resource 304. If so, a local function is called 312 and control is returned 310 to the application. If it is not a local resource, the system creates a response signal 306. If the response signal 306 cannot be created, control is returned to the application. If it is, then the local list is searched 314 for the communication handle. If the communication handle is not found 316, a communication handle is obtained 318 from the requester and then ownership if the write token is requested 320. However, if the communication handle is found 316, then ownership if the write token is immediately requested 320.

If no error occurs when the request for ownership of the write token is made 322, then the response signal is stored in the packet header 326, a request packet is built into the write token 328, the write thread sends the packet, and the write token is released 332. If an error is detected when the packet is sent, the response signal is destroyed 342 and control is returned 344 to the application. If no errors occur during packet transmission 344, then the system waits 336 for the response packet, the data in the response packet is transferred 338 into the application's buffer, the read token is released 340, the response signal is destroyed 342 and control is returned 344 to the application.

FIGS. 4A–C illustrate the memory layout of the packets used in the preferred embodiment. FIG. 4A illustrates a packet as encrypted by security level 1. In security level 1, the packet header is encrypted using single DES encoding. This level of security incurs the least amount of overhead and is preferably used in more secure environments such as LANs.

FIG. 4B illustrates a packet as encrypted by security level 2. In security level 2, the packet header and data are encrypted using single DES encoding. This level of security incurs slightly increased overhead as compared to security level 1, but provides an increased level of security for less secure environments such as wide area networks.

FIG. 4C illustrates a packet as encrypted by security level 3. In security level 3, the packet header and the data are encrypted using triple DES encoding. This level of security incurs the most overhead as compared to security levels 1 and 2, but provides the highest level of security for insecure environments such as public telephone networks.

To protect data exchanged over communication sessions, the preferred embodiment provides two different encryption schemes available to the user at logon. The first scheme is the US Department of Defense Data Encryption Standard (DES) and the second scheme is the triple-DES specified in the ANSI X9.17 and ISO 8732 standards but with three different keys. In addition, the preferred embodiment applies the Cipher Block Chaining mode specified in the FIPS PUB 81 to better protect the data. Once an encryption scheme is selected, data exchanged over all sessions connected to a network domain are encrypted regardless of the communication protocols being used by the sessions. The price to paid for the encryption is minimum anyway since the preferred embodiment encrypts 500,000 bytes per second when running on a Pentium 66 MHz processor. The operating system used can be any suitable personal computer operating system such a Microsoft (TM) Windows 95 (TM), IBM (TM) OS/2 Warp (TM), Unix, etc. If the server is a large system, any one of a number of suitable mainframe operating system software may be used.

In addition to the above encryption schemes, the preferred embodiment employs a dynamic packet header technique to provide extra securities based on the security level selected by the user at logon. If a security level 2 is selected, the packet header and data are encrypted with DES and the packet header is changed to 24 bytes to carry the CRC signatures of the packet header and data for authentication. However, if a security level 3 is selected, the packet header and data are encrypted with triple-DES using three different keys. Finally, if security level 1 is selected, the packet header remains at 16 bytes and no signature is verified for a better performance but the packet header is encrypted with DES to provide security against other threads. Thus, thanks to the dynamic packet header technique, a user can setup different types of firewalls wherever he needs them. For instance, the user can connect to his office from his home using security level 2 and setup his office machine to connect to another server within his organization using a lower security level to gain a better performance.

In order to provide better security, the preferred embodiment allows the user to select if the data should stay in its encrypted form so that only authorized personnel can view the data. This is important for sensitive business data, personnel data, etc. Of course, the key to decrypt the data must be agreed to ahead of time or exchanged over some secured channels to protect the secrecy of the key.

Of course, those skilled in the art will recognize that the user could also have the capability of instructing the system that no encryption will be used. In this case, no encryption would represent a fourth security level (security level 0). Security level 1–3 having been discussed in regard to FIG. 4.

FIGS. 5A–B illustrate the packet queue structure used in the preferred embodiment. FIG. 5A illustrates the TCP/IP and NetBIOS communications structure and FIG. 5B illustrates the SMODEM and SRS232 communications structure. The compressed buffer is a work buffer used to compress data prior to transmission through SMODEM or SRS232 communication lines. A packet header is placed at the beginning of the read token and at the beginning of the write token. In the preferred embodiment, the read and write tokens are stored in shared memory.

FIG. 6 illustrates a configuration in which multiple requesters 110 communicate with a single server 602.

FIG. 7 illustrates a configuration in which a single requester 110 communicates with multiple servers 602.

FIG. 8 illustrates a configuration in which a system 802 and multiple servers 804 communicate with one another.

FIG. 9 illustrates a configuration in which multiple systems 802 and multiple servers 804 communicate with one another via modems 124 over phone lines 906 and also over LANs 902 and wide area networks 904. This figure illustrates the ability of the system to interface with multiple communications protocols.

FIG. 10 illustrates a configuration in which multiple requester systems 1002, multiple server systems 1004, and multiple server/requester systems 1006 communicate with one another. The configuration in this figure is similar to that shown in FIG. 9.

FIGS. 11 and 12 illustrate a configuration in a server 1004 which includes communication sessions 1120 to communicate with requesters, encrypter/decrypter 1128, read threads 1114, write threads 1116, packet queues 1110, 1112, a resource control manager 1102 to control user ID, access permission and alias and path storage 1104, 1106, 1108. The cached user ID and access permission 1124 and the cached alias and associated path 1126 caches are used to store data from the access permission storage 1106 and the alias and path storage disks 1108 for improved system performance.

To protect resources on the network domains, an access control list (ACL) is used for each network domain in access permission storage 1106. The ACLs are managed by network administrators to define to which resources a user can access and what kind of accesses the user has to each resource. The system provides a sophisticated ACL so that a user cannot view or access any resources other than those assigned. The following access permissions are used by our ACLs:

READ_FILE
WRITE_FILE
CREATE_FILE
DELETE_FILE
EXECUTE_FILE
CHANGE_ATTRIBUTE
ACCESS_SUBDIR
CREATE_SUBDIR
REMOVE_SUBDIR

For example, if the user is not permitted access to any subdirectories from a network resource, the user will not see any subdirectory at all when viewing the network resource. If for some reasons the user knows a particular subdirectory exists under the network resource, he cannot access it anyway. The management of network resources and user access permissions is provided with a user-friendly Graphical User Interface application. Together with the logon procedure, ACLs provide effective protections to the resources on the network domains.

FIG. 12 is a more detailed view of the server 1004 of FIG. 11. A control manager 1122 within the server 1004 is responsible for communication between the server 1004 and other applications on the server 1004 machine. Thus, the server 1004 can be informed if a database has been changed by a resource control application. The server 1004 can also accept a message from another application 102 to send to all or selected clients over active sessions. If an electronic mail system should be needed, the server 1004 can save the message and wait until a client is logged on to send the message over the session. To support these features, the control manager 1122 posts message or e-mail packets to the incoming packet queues 1206 of the sessions 1120. When the server processing threads 1114, 1116 of the sessions 1120 retrieves the packets from the queue 1206, it will process the packets based on the packet types defined in the packet headers.

FIG. 13A–D illustrates the packet headers used in the logon procedure. A session key KS and an initialization vector IV are defined for a communication session between a client and a server 1004 when security level 1 or higher is desired (in security level 0, no encryption is used).

FIG. 13E illustrates a normal packet such as those used during data transfer. When an e-mail or message packet is sent, the preferred embodiment uses security level 2 by default to protect the messages. In security level 2, both packet header and data are encrypted using single DES encryption.

The requester also has the capability to signal request routers 106 of all applications 102 when a communication session is terminated abnormally whether the request routers 106 are sending request packets or waiting on response packets. In order to perform this feature, the response signals (i.e., the return addresses stored in the request packets) are saved in response-signal queues by the session write thread 1116. Each communication session has a response-signal queue 1206 to reduce the search time. When the response packets are successfully delivered, their corresponding response signals are removed from the queue by the session read threads 1114 of the corresponding communication channels. If an application 102 terminates before its response packets arrive, the response packets are discarded and the response signals are also removed from the queue after all chaining response packets have arrived.

In addition, the read thread of the client session also recognizes different types of packets to determine whether it should route the received packets to the application's request router or to a message manager within the requester. The message manager of the requester is responsible for message and e-mail packets sent from the connected servers. This feature is important because it allows the server to initiate the sending of packets while a session is active. As an example, a hot-link can be defined so that a server can inform the connected clients if a database should be changed or a server administrator can send a message to all or selected clients telling them if a server should be out of service shortly, etc. In a more advanced application, an electronic-mail server application can be written so that the message packets are saved on the server until a client is logged on. At that time, the server will send the saved messages to the connected client.

In the prior art, the requester is the one that translates and formats requests from the applications; thus, it cannot perform preparation ahead of time. In addition, information accumulating in one place could increase the search time. The prior art requires its intrinsics modules in both the application and the requester which may require more resources to be allocated and more machine instructions to be executed. Furthermore, the prior art does not have the capability to accumulate multiple request packets from a requester so that the server can process the next packet request while the previous response packet is traveling back to the requester on the network or being processed by communication devices in their own memory buffers.

In contrast to the prior art, the preferred embodiment contains the formatting and translating code in just one place, the request router 106. Our requester only encrypts packet headers and packet data if necessary and then calls the transport functions to send the packets to the server. In addition, requester 110 is also responsible for saving logon and mounting information, managing the communication sessions, and delivering response packets received from multiple network domains to multiple request routers while sending request packets to the multiple network domains. Requester 110 does not need to know the format of the response data, and can deliver the response packets immediately upon receiving them. The request routers 106 can then format or translate the response data in the applications timeslices while the requester 110 is waiting for other incoming response packets or reading data from the communication devices 120, 122, 124, 126. Thus, the preferred embodiment achieves better performance than the prior art.

The prior art also requires the intrinsic modules to translate and format the application data from a program stack segment to a parameter block before sending it to its requester where the data Is once again formatted or copied into a data communication buffer. In contrast, the request routers 106 in the preferred embodiment format the application data only once and store the formatted data into the write token which will be used by the requester and the communication subsystem to send the request packets to the server. When the response packets arrive, the requester 110 uses the response signals to tell the corresponding request routers that their response packets have arrived. At that time, the request routers 106 transfer response data directly from the read tokens into the application buffers. Thus, the preferred embodiment eliminates the overhead of copying data between memory buffers.

Furthermore, the prior art does not have the dynamic packet header feature to support packet authentication on demand. Neither does its server authenticate the requester to prevent replaying of packets by intruders. The prior art also requires two different programs running on the server to wait for incoming data from different communication protocols. The preferred embodiment only requires the server to be started once for multiple communication protocols.

In general, a session on the server 1004 will support multiple applications on the requester; thus, a server 1004 must somehow remember the resources allocated for the client applications so that these resources can be released whether the client applications terminate abnormally or the communication sessions are destroyed abnormally. Our server supports this feature in each session thread. Since the allocated resources are isolatedly remembered for different requesters, the search time is minimum every time they are added or removed from the memorized list. In addition, security audit can be turned on and off by the network resource manager running on the server over the control channel of the server. The network resource manager can toggle the security audit for users or groups whose IDs are supplied in the auditing request packet, or resources whose names are stored in the auditing request packet. The audit can also be logged based on successful, failed, or both transactions.

In the prior art, the application is the one which determines if a session should be started on the host computer. The application then makes a function call to connect to the host computer and another function call to start a host server process. In the preferred embodiment, the session manager of the requester determines if a connection should be established to couple the client computer to the server computer. Once the connection is established, the server automatically creates a server processing thread to process the client request packets received over the connection. After the connection is established, the session manager also performs the auto-logon itself, not the application. The session can then be shared by all the applications on the client machine.

Thus, the session creation and logon are transparent to the applications. If the logon is successful, the server creates a server receiving thread to receive and accumulate request packets in a packet queue so that they will be processed by the server processing thread. When a session disconnect request packet is received, the server receiving and processing threads terminate themselves. However, if the communication session is destroyed abnormally, the server receiving thread simulates a disconnect request packet and appends it to the packet queue to signal the server processing thread to terminate. The server receiving thread then terminates itself.

Note that in the very first logon manually performed by the user, the operation is slightly different than the auto-logon mentioned in the above paragraph. The requester first receives a logon request from the logon application, it establishes the session itself and then performs the logon. This is so done by the command manager of the requester, not by the session manager.

Since request packets are accumulated in the packet queue in the preferred embodiment, the request packets may not be processed immediately upon arrival. In contrast, the prior art must process the request packets immediately to return the status or data to the requester. This may indicate that other applications on the client computer must wait until the return packet has arrived and processed before they can send their requests to the same host computer.

The prior art requires an application to send a function call to the host computer to established a communication session. Our system establishes a communication session by the requester when it receives a logon request from the logon program or a request router asking for the communcation handle. In addition, our server has the capability to reformat and retranslate the request packets in its own request router before forwarding them to the requester located on the server when the network resources do not reside on the server. That is, multiple servers can be connected together as shown in FIGS. 7–10 to expand the amount of network resources available to requesters. Note that this feature requires the intermediate servers' administrator(s) to manually logon the designated servers since the logon passwords are not stored on the intermediate servers. Users on requesters can perform this logon remotely if their access permissions in the ACLs of the intermediate servers indicate that they can execute programs on the intermediate servers. However, caution must be taken and security level 3 is advised when using this feature since logon user IDs and passwords must be sent along with the executing request packets.

As shown earlier, the very first logon packet is encrypted with three different keys for different parts of the packet. The header of the logon packet is encrypted with a key generated from the server name. This is design to detect outside intruders early in the verification process. For intruders working inside an organization, the server name may be known. Then it comes the middle part of the logon packet which contains the 64-bit random number and the CRC values. These are the heart of the verification since it is encrypted with the key generated from the user ID and the secret password. This scheme allows the server to detect the intruding logon right on the very first packet. The challenge-response process that following the logon packet is to defeat re-played packets.

The encryption system used in the preferred embodiment has several other advantages, as follows. The long term key is derived from a user ID and a secret password. It has 192 bits and is used in a triple-DES encryption enhanced with CBC. The short term key is generated with the X9.17 key generation formula and changed every time a session is established between two nodes on the network. Thus, the encryption occurs at the application layer which exposes the source and destination addresses of the packets when used with TCP/IP and NetBIOS protocols but the intruders must deal with different keys whose lengths are either 64 or 192 bits for different pair of nodes on the network. In addition, the short term key is encrypted and only sent once when the communication session between two nodes is established, not in every packet; thus, it reduces the traffic between two nodes.

Furthermore, the prior art only protects data between site-firewalls, not between nodes. In many cases, data must be protected between nodes within an organization. For instance, high-rank management officers within a private network may want to exchange restricted confidential information without leaks to their employees.

Encryption at the application layer also reduces the cost of replacing the existing network layer and can be done on demand when protection to data is needed. Different security firewalls can easily be established between any pair of nodes with a single click of the fingertip.

Finally, the communication subsystem of the preferred embodiment is a foundation for multiple applications when their use are in demand. With just one communication session between a client and a server, packet sending can be initiated by either party to conduct file transfers, broadcast messages, or e-mail messages. In addition to minimum resources and maximum performance, security is also provided to protect the secret of the data.

While the invention has been described with respect to a preferred embodiment thereof, it will be understood by those skilled in the art that various changes in detail may be made therein without departing from the spirit, scope, and teaching of the invention. For example, the size of encryption keys can be changed, algorithms used to generate the encryption keys can be changed, the device can be implemented in hardware or software, etc. Accordingly, the invention herein disclosed is to be limited only as specified in the following claims.

I claim:

1. A bi-directional security system for a network, comprising:

at least one client, the client further comprising:

client communication means to communicate with at least one server;

packet reception means to receive transmitted packet data from the server;

means to generate and transmit a first packet to the server, at least a portion of the first packet having a first packet header containing client identifying information;

means to encrypt at least a portion of the client identifying information in the first packet header prior to transmission;

means to decrypt at least a portion of the client authenticating information in a second packet header and to determine if the second packet is from the server, the client further having means to terminate the communication if the second packet is from an invalid server;

means to generate and transmit a third packet to the server, at least a portion the third packet having a third packet header containing session information; and

means to encrypt at least a portion of the session information in the third packet header prior to transmission; and

the server further comprising:

server communication means to communicate with the client;

packet reception means to receive transmitted packet data from the client;

means to decrypt at least a portion of the client identifying information in the first packet header and to determine if the first packet is from a valid client, the server further having means to terminate the communication if the first packet is from an invalid client;

means to generate and transmit a second packet to the client in response to the first packet, at least a portion the second packet having the second packet header containing client authenticating information;

means to encrypt at least a portion of the client authenticating information in the second packet header prior to transmission; and

means to decrypt at least a portion of the session information in the third packet header;

whereby, the client and the server each verify the validity of the other by transmitting encrypted identifying information to one another.

2. A security system, as in claim 1, further comprising:

means in the server to generate and transmit a fourth packet to the client in response to the third packet, the fourth packet having a packet header containing session information; and

means to encrypt at least a portion of the session information in the fourth packet header prior to transmission.

3. A security system, as in claim 2, wherein:

the client has a userid;

the client has a password;

the first packet is encrypted by:

concatenating a random number to a predetermined bit constant to form a value R;

a CRC signature C1 is generated from the value R and the userid;

the value R is used as a DES key to encrypt the userid;

the server name is used to generate a key K to encrypt the value R;

the key Ka is generated by a one way hash function from the userid and password; and

a random number Ra and its CRC signature C2 is generated, Ra and C2 are encrypted using key Ka.

4. A security system, as in claim 3, wherein:

the server further comprises an encrypted client password file;

the second packet is encrypted by:

a key K2 is generated from the server name and a one way hash function to decrypt the packet header of the first packet;

the userid is decrypted using the decrypted value R from the packet header;

the decrypted userid is used to access an authorization table to determine if the first packet is valid;

the userid is used to extract a one way hashed password Kb from the encrypted client password file, the password Kb is then used to decrypt values Ra, C1 and C2;

the value Ra is manipulated via a predetermined formula to produce a random number R'a;

a random number Rb is generated by the server; and

R'a and Rb are encrypted with password Kb, inserted into the packet header of the second packet and transmitted to the client.

5. A bidirectional security system for a network, comprising:

at least one client, the client further comprising:

means to encrypt a first logon packet;

means to transmit the first logon packet to the server;

means to decrypt the second logon packet;

means to encrypt a third logon packet with session information;

a server, further comprising:

means to decrypt the first logon packet;

means to encrypt a second logon packet with client authenticating information;

means to transmit the second logon packet to the client;

means to decrypt the third logon packet; and

a communication channel capable transmitting packets between the client machine and the server;

whereby the client and server can establish secure communications by bi-directionally transmitting encrypted data.

6. A security system, as in claim 5, further comprising:

means to encrypt packet data in least two security levels, the first security level having a first packet encryption scheme and the second security level having a second packet encryption scheme;

whereby the security system can selectably encrypt packet data with at least two packet encryption schemes.

7. A security system, as in claim 6, further comprising:

means to encrypt packet data at least three security levels, the third security level having a third packet encryption scheme;

whereby the security system can selectably encrypt packet data with at least three packet encryption schemes.

8. A security system, as in claim 7, wherein the first packet encryption scheme is a single DES encryption.

9. A security system, as in claim 8, wherein the second packet encryption scheme is a triple DES encryption.

10. A security system, as in claim 9, wherein:

the first packet encryption scheme encrypts the packet header information; and

the second packet encryption scheme encrypts the packet header information;

the third packet encryption scheme is a triple DES encryption, and further encrypts the packet header and the packet data.

11. A security system, as in claim 10, wherein:

the server further comprises means to encrypt a fourth logon packet with session information; and

the client further comprises means to decrypt the fourth logon packet.

12. A security system, as in claim 9, wherein:

the client further comprises means to encrypt data packets; and

the server further comprises means to encrypt data packets;

data packets are selectably encrypted using at least one of the security levels; and

means to dynamically adjust the size of the packet header based on the selected encryption scheme.

13. A security system, as in claim 5, wherein:

each client includes at least one application program; and

the server further comprises at least one packet queue for each client;

whereby application performance is improved by reducing packet search time.

14. A method of securely transmitting packet data between a client and a server with encrypted packets, including the steps of:

using at least one communication channel to transmit packets between at least one client machine and at least one server;

encrypting in the client a first logon packet;

transmitting the first logon packet to the server;

decrypting the first logon packet in the server;

encrypting a second logon packet in the server with client authenticating information;

transmitting the second logon packet to the client;

decrypting the second logon packet in the client;

encrypting in the client a third logon packet with session information;

decrypting the third logon packet in the server;

whereby the client and server can establish secure communications by bi-directionally transmitting encrypted data.

15. A method, as in claim 14, including the further steps of:

encrypting a fourth logon packet in the server with session information;

transmitting the fourth logon packet to the client; and

decrypting the fourth logon packet in the client;

using the session information to control encryption of packets while communicating between the client and the server.

16. A method, as in claim 15, including the further step of using at least two selectable encryption schemes, including a first encryption scheme for a first security level and a second encryption scheme for a second security level.

17. A method, as in claim 16, including the further steps of:

using at least two communication channels to communicate between multiple client and server, at least a first communication channel having a first level of security and at least a second communication channel having a second level of security; and

selecting the first encryption scheme for the first communication channel and the second encryption scheme for the second communication channel.

18. A method, as in claim 17, including the further step of using single DES encryption for the first level of security and triple DES encryption for the second level of security.

19. A method, as in claim 18, including the further steps of:

using packets which contain a header portion and a data portion; and

using a third encryption scheme in which triple DES encryption is used for the packet header and the packet data.

20. A method, as in claim 19, including the further steps of:

selecting the encryption scheme based on the nature of the data in the packet; and

dynamically adjusting the size of the packet header based on the selected encryption scheme.

* * * * *

US006510349B1

(12) United States Patent

Schneck et al.

(10) Patent No.: US 6,510,349 B1

(45) Date of Patent: *Jan. 21, 2003

(54) ADAPTIVE DATA SECURITY SYSTEM AND METHOD

(75) Inventors: **Phyllis A. Schneck**, Potomac, MD (US); **Karsten Schwan**, Tucker, GA (US); **Santosh Chokhani**, Arlington, VA (US)

(73) Assignee: **Georgia Tech Research Corporation**, Atlanta, GA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/528,759**

(22) Filed: **Mar. 17, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/181,304, filed on Oct. 28, 1998, now Pat. No. 6,108,583

(60) Provisional application No. 60/063,551, filed on Oct. 28, 1997.

(51) **Int. Cl.**^{7} **G05B 15/02**

(52) **U.S. Cl.** **700/9**; 700/7; 700/8; 700/67; 700/68; 709/213; 709/229; 709/230; 713/186; 713/187; 713/188; 370/246; 370/352; 370/389; 380/29; 380/30; 380/274; 380/286

(58) **Field of Search** 700/2, 7–9, 68, 700/67; 713/153–156, 162–168, 181–186, 200–201, 202, 203, 191; 714/713; 370/352, 246, 389, 1–2; 380/26–29, 30, 274–278, 286, 44–47; 704/213, 229, 230

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,771,391	A	9/1988	Blasbalg	364/514
4,965,827	A	10/1990	McDonald	380/25
5,005,137	A	4/1991	Ernst	364/514

(List continued on next page.)

OTHER PUBLICATIONS

Meyer, et al., "MPEG–FAQ 4.1," Security for Multimedia–Data with the Example MPEG–I–Project, Project Description of SECMPEG, 1995, pp. 1–2.

Jha, et al., "Adaptive Resource Allocation for Embedded Parallel," Proceedings of Third International Conference on High Performance Computing, Dec. 1996.

(List continued on next page.)

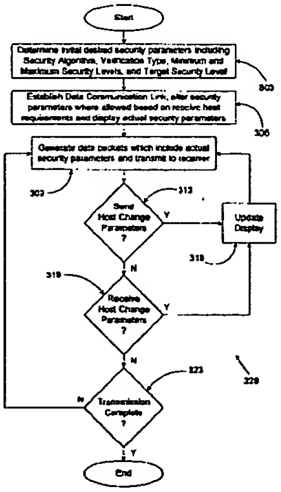
Primary Examiner—Ramesh Patel

(74) *Attorney, Agent, or Firm*—Thomas, Kayden, Horstemeyer & Risley

(57) ABSTRACT

A system and method for data communication with adaptive security in which a send host transmits a data stream to a receive host in packets which contain an authentication data block with an authentication header and a signature block. The authentication header advantageously contains various fields including a verification type, a security algorithm, a minimum security level, a target security level, and an actual security level. The receive host adaptively performs verification of the data packets using varying security levels based in part on the availability of security operations per second (SOPS) in the receive host. Where a data stream in the receive host is delayed by a security processing bottleneck, the receive host may alter the verification type, security algorithm, or the actual security level to speed up the processing of the data stream by reducing the amount of security processing performed. The receive host further allocates the SOPS among the data streams received based on a priority assigned to each data stream.

31 Claims, 8 Drawing Sheets


Start
Determine initial desired security parameters including Security Algorithm, Verification Type, Minimum and Maximum Security Levels, and Target Security Level
303
Establish Data Communication Link, alter security parameters where allowed based on receive host requirements and display actual security parameters
306
Generate data packets which include actual security parameters and transmit to receiver
309
313
Send Host Change Parameters ?
Y
Update Display
316
N
319
Receive Host Change Parameters ?
Y
N
323
329
N
Transmission Complete ?
Y
End

U.S. PATENT DOCUMENTS

5,107,415	A		4/1992	Sato et al.	395/800
5,477,531	A		12/1995	McKee et al.	370/17
5,481,735	A		1/1996	Mortensen et al.	395/200.1
5,490,252	A		2/1996	Macera et al.	395/200.01
5,511,122	A		4/1996	Atkinson	380/25
5,541,852	A		7/1996	Eyuboglu et al.	364/514 C
5,550,984	A		8/1996	Gelb	395/200.17
5,566,310	A		10/1996	Mathewson, II	395/200.13
5,586,260	A		12/1996	Hu	395/200.2
5,627,970	A		5/1997	Keshav	395/200.13
5,627,972	A		5/1997	Shear	395/200.18
5,642,360	A		6/1997	Trainin	370/230
5,715,397	A		2/1998	Ogawa et al.	395/200.18
5,727,159	A		3/1998	Kikinis	395/200.76
5,737,535	A		4/1998	Bagley et al.	395/200.57
5,761,438	A		6/1998	Sasaki	395/200.77
5,764,921	A		6/1998	Banham et al.	395/200.77
5,768,536	A		6/1998	Strongin et al.	395/200.77
5,784,571	A		7/1998	Mantopoulos et al.	395/200.77
5,784,572	A		7/1998	Rostoker et al.	395/200.77
5,784,578	A		7/1998	Galloway et al.	395/285
6,108,583	A	*	8/2000	Schneck et al.	700/67
6,115,376	A	*	9/2000	Sherer et al.	370/389
6,219,771	B1	*	4/2001	Kikuchi et al.	711/164
6,240,184	B1	*	5/2001	Huynh et al.	380/206

OTHER PUBLICATIONS

Ivan–Rosu, et al., "Improving Protocol Performance by Dynamic Control of Communication Resources," Proceedings of 2nd IEEE Conference on Complex Computer Systems (ICECCS), Oct. 1996.

Rosu, et al., "On Adaptive Resource Allocation for Complex Real-Time Applications," Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS), IEEE, 1997.

Schneck, et al., "DRRM: Dynamic Resource Reservation Manager," International Conference on Computer Communications and Networks, IEEE, 1996.

Waldspergcr, et al., "Lottery Scheduling: Flexible Proportional–Share Resource Management," Operating Systems Review, 1994, pp. 1–12.

Agi, et al., "An Empirical Study of Secure MPEG Video Transmissions," IEEE, Mar. 1996, pp. 1–8.

Li, et al., "Security Enhanced MPEG Player," Department of Computer Science, University of Illinois at Urbana–Champaign, 1996.

Nahrstedt, et al., "Resource Management in Networked Multimedia Systems," Handbook of Multimedia Networking, Auerbach Publications, 1995, pp. 381–405.

Needham, et al., "Using Encryption for Authentication in Large Networks of Computers," Communications of the ACM, vol. 21, No. 12, Dec. 1978, pp. 210–216.

Reiter, et al., "A security Architecture for Fault–Tolerant Systems," ACM Transactions on Computer Systems, vol. 12, No. 4, Nov. 1994, pp. 340–371.

Larry L. Peterson, "OS Support for High–Speed Networking," http://www.cs.arizona.edu/xkernel/projects/cryptography–and–protocols, pp. 1–11.

Hudson, et al., "Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems," Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work, pp. 248–257.

Varadharajan, et al., "A Practical Approach to Design and Management of Secure ATM Networks," Jul. 1997 pp. 213–232.

Rumen Stainov, "Dynamic Protocol Configuration for Multimedia Networks," Microprocessing and Microprogramming, 38, 1993, pp. 741–748.

* cited by examiner

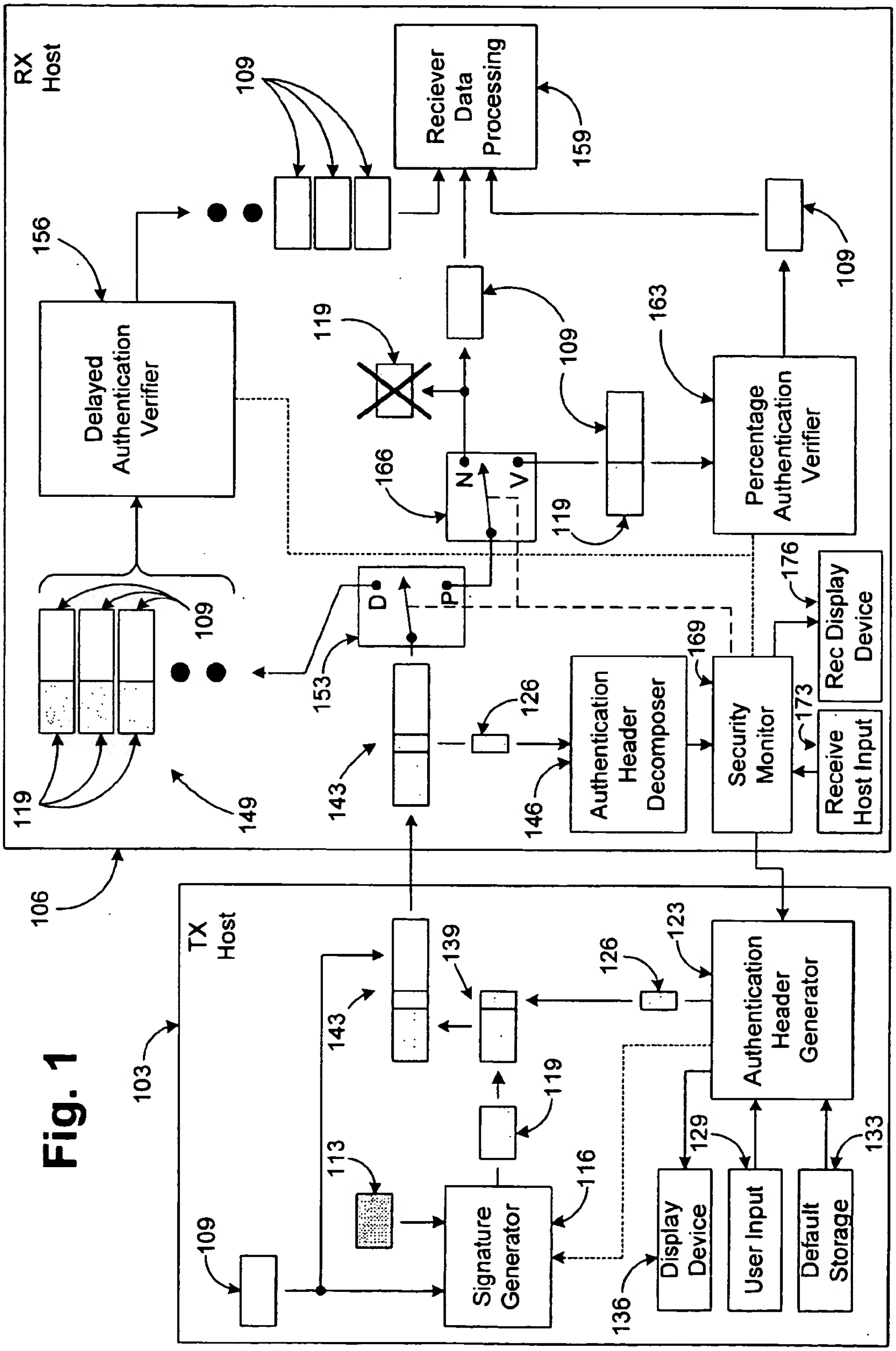

RX Host
Reciever Data Processing
109
159
156
Delayed Authentication Verifier
119
109
163
166
N
V
Percentage Authentication Verifier
119
109
D
P
153
176
Rec Display Device
169
Security Monitor
173
Receive Host Input
Authentication Header Decomposer
146
126
143
149
119
106
TX Host
103
143
139
126
123
Authentication Header Generator
119
113
116
109
Signature Generator
129
133
136
Display Device
User Input
Default Storage

Fig. 1

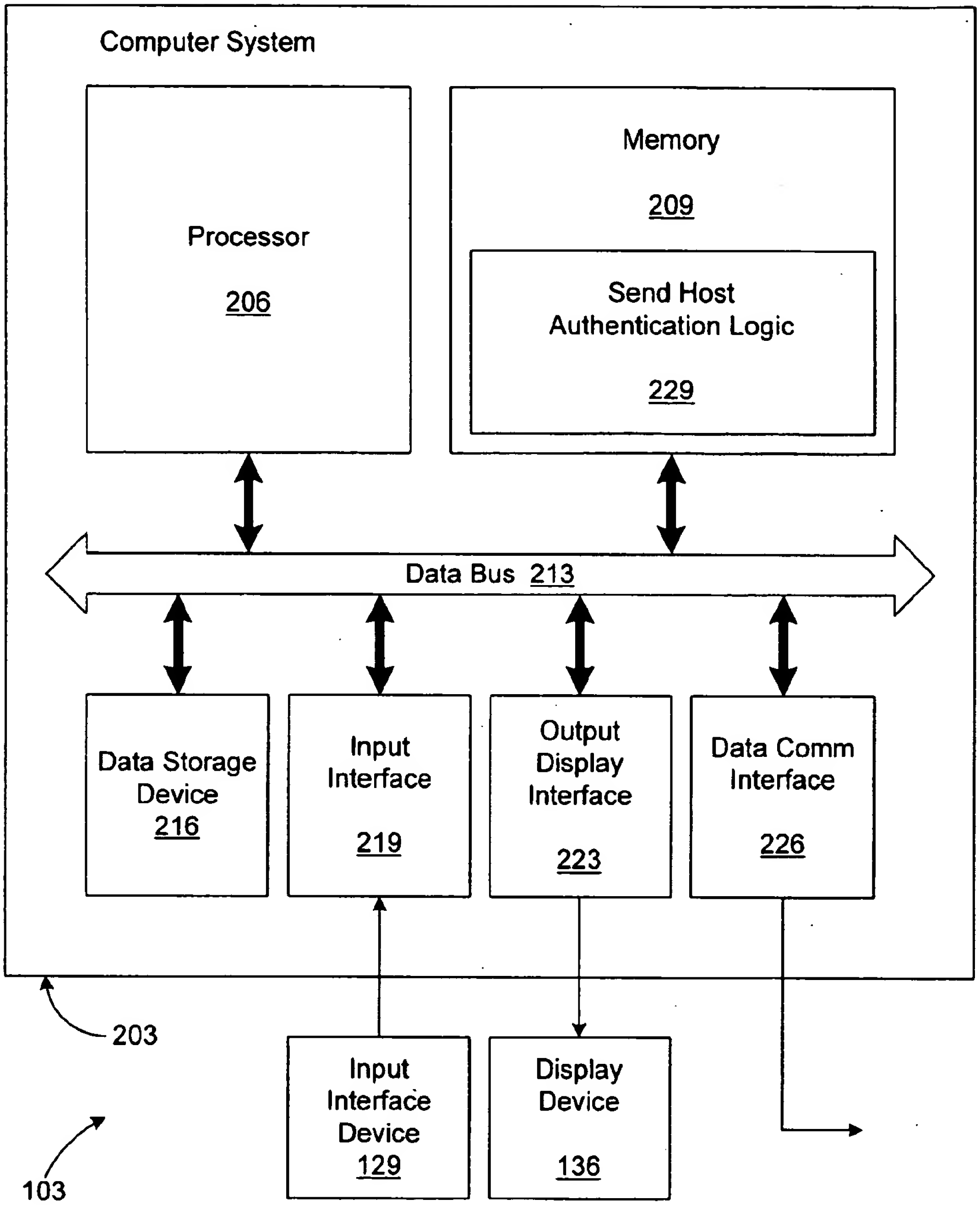

Computer System
Processor
206
Memory
209
Send Host Authentication Logic
229
Data Bus 213
Data Storage Device
216
Input Interface
219
Output Display Interface
223
Data Comm Interface
226
203
Input Interface Device
129
Display Device
136
103

Fig. 2

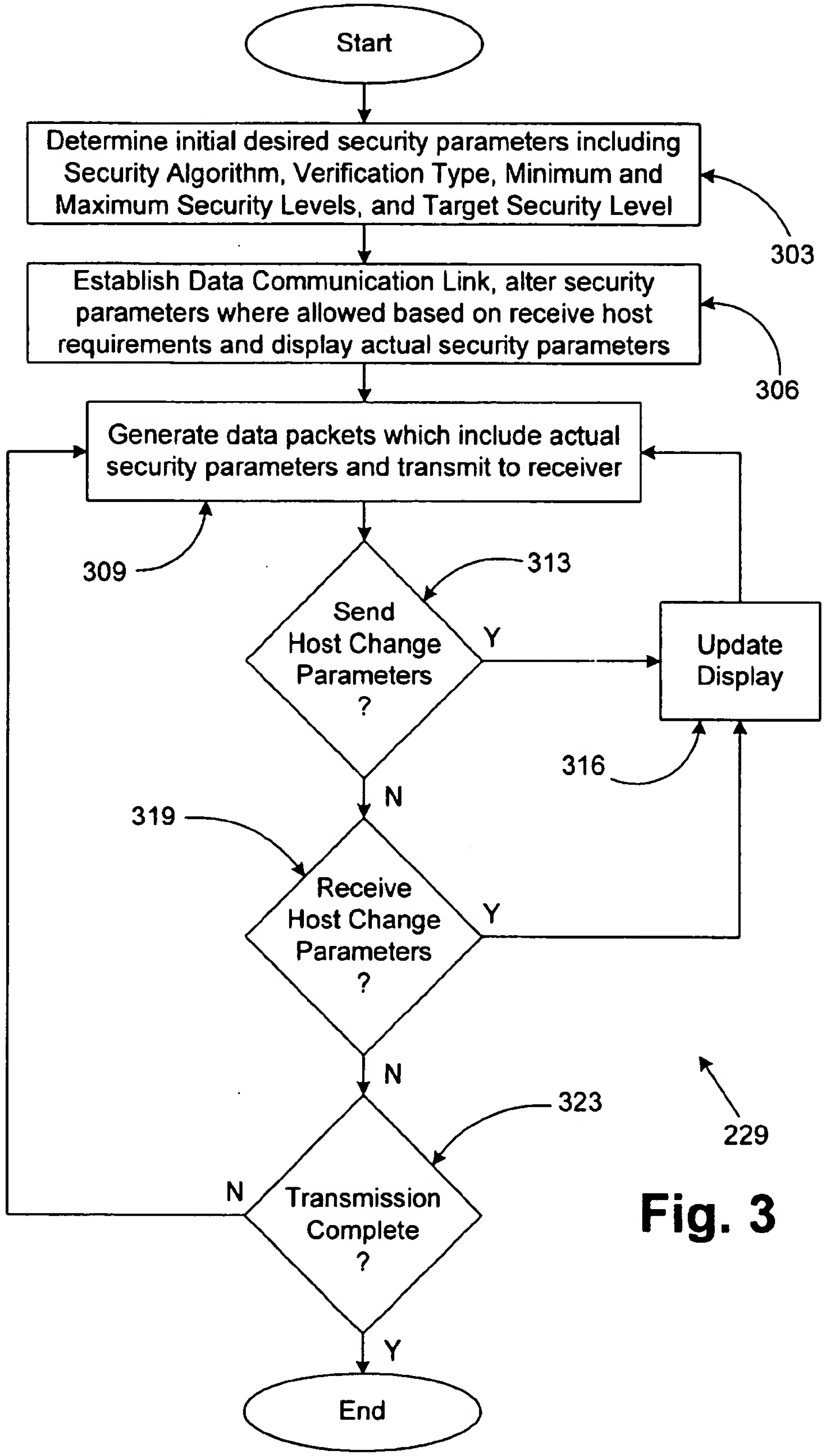
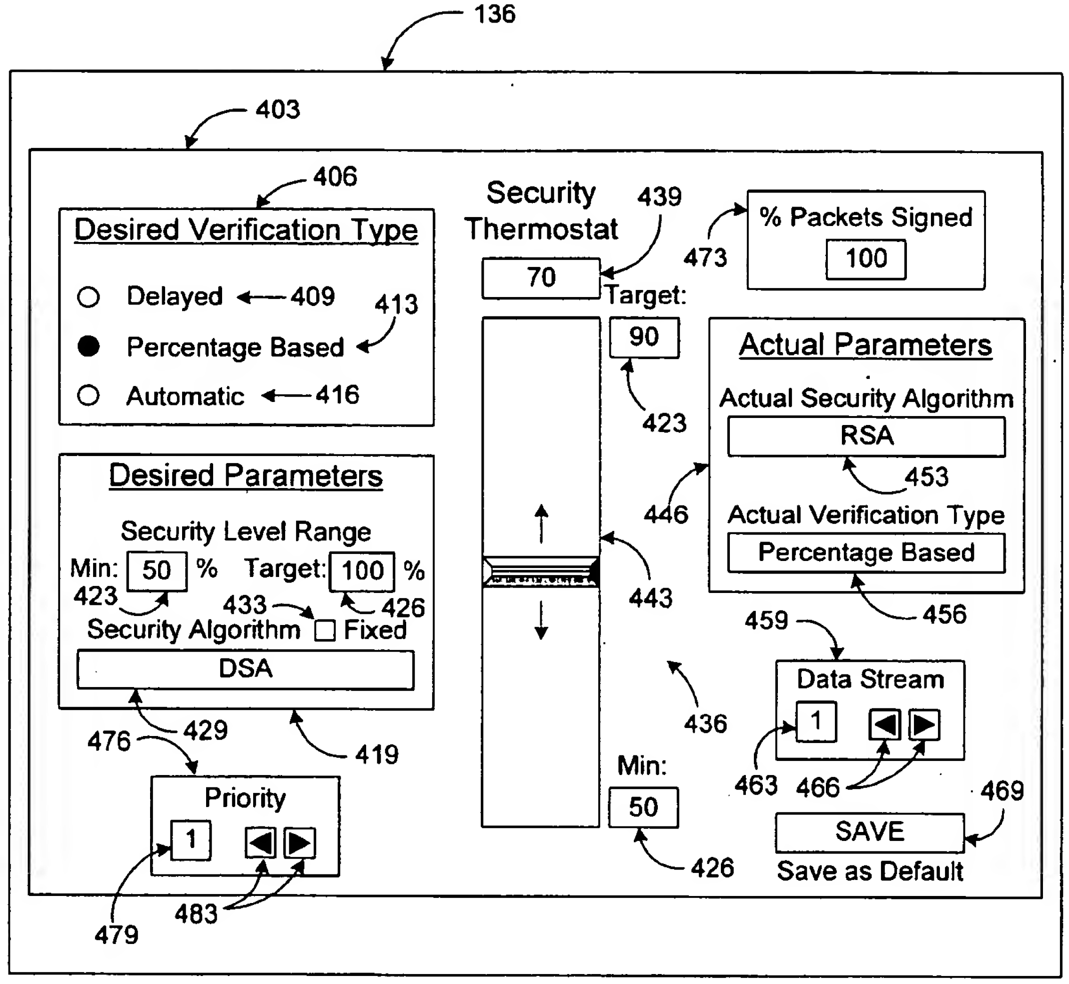

Start
Determine initial desired security parameters including Security Algorithm, Verification Type, Minimum and Maximum Security Levels, and Target Security Level
303
Establish Data Communication Link, alter security parameters where allowed based on receive host requirements and display actual security parameters
306
Generate data packets which include actual security parameters and transmit to receiver
309
313
Send Host Change Parameters ?
Y
Update Display
316
N
319
Receive Host Change Parameters ?
Y
N
323
229
N
Transmission Complete ?
Y
End

Fig. 3



Desired Verification Type

○ Delayed

● Percentage Based

○ Automatic

Desired Parameters

Security Level Range

Min: 50 % Target: 100 %

Security Algorithm □ Fixed

DSA

Priority

1 ◀ ▶

Security Thermostat

70

Target: 90

Min: 50

% Packets Signed

100

Actual Parameters

Actual Security Algorithm

RSA

Actual Verification Type

Percentage Based

Data Stream

1 ◀ ▶

SAVE

Save as Default

Fig. 4

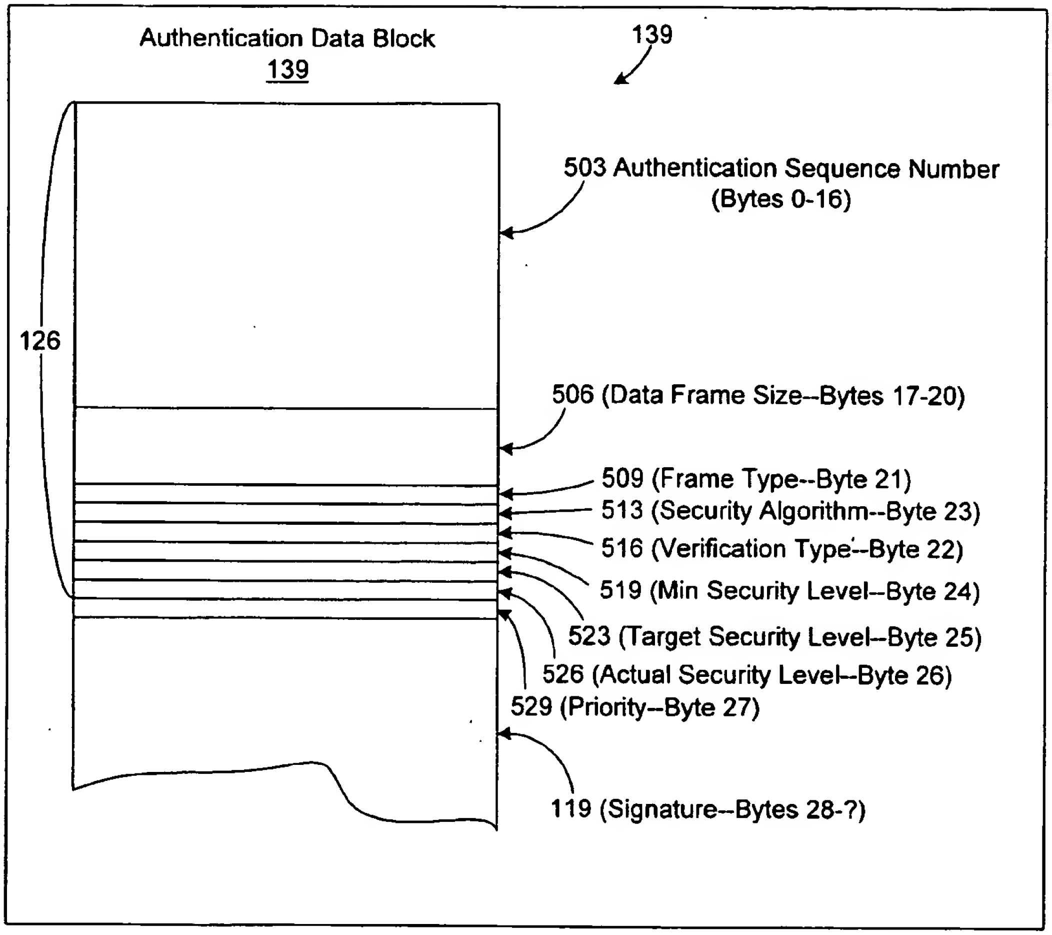

Authentication Data Block
139
139
503 Authentication Sequence Number (Bytes 0-16)
126
506 (Data Frame Size--Bytes 17-20)
509 (Frame Type--Byte 21)
513 (Security Algorithm--Byte 23)
516 (Verification Type--Byte 22)
519 (Min Security Level--Byte 24)
523 (Target Security Level--Byte 25)
526 (Actual Security Level--Byte 26)
529 (Priority--Byte 27)
119 (Signature--Bytes 28-?)

Fig. 5

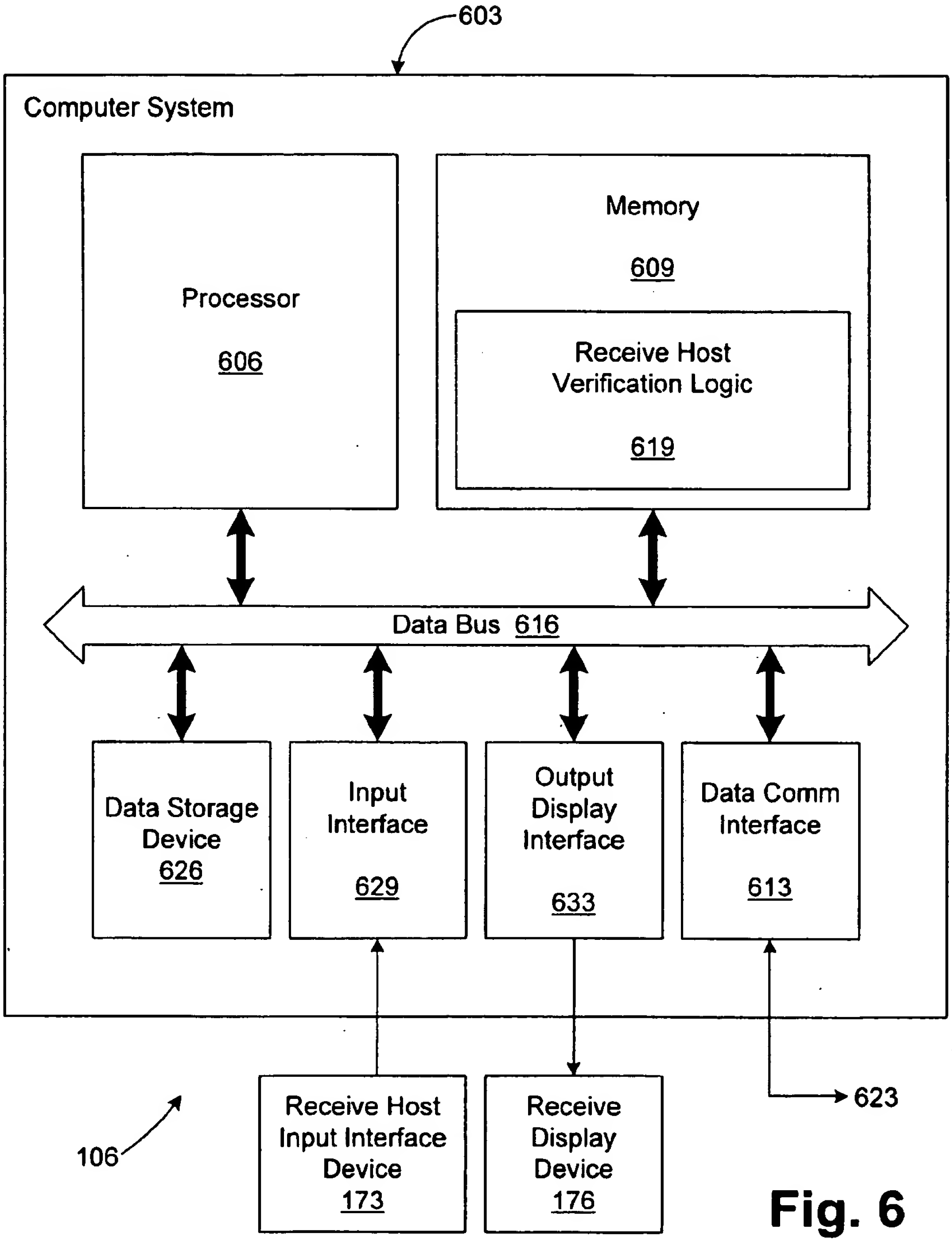

603
Computer System
Processor
606
Memory
609
Receive Host
Verification Logic
619
Data Bus 616
Data Storage Device
626
Input Interface
629
Output Display Interface
633
Data Comm Interface
613
Receive Host Input Interface Device
173
Receive Display Device
176
623
106

Fig. 6

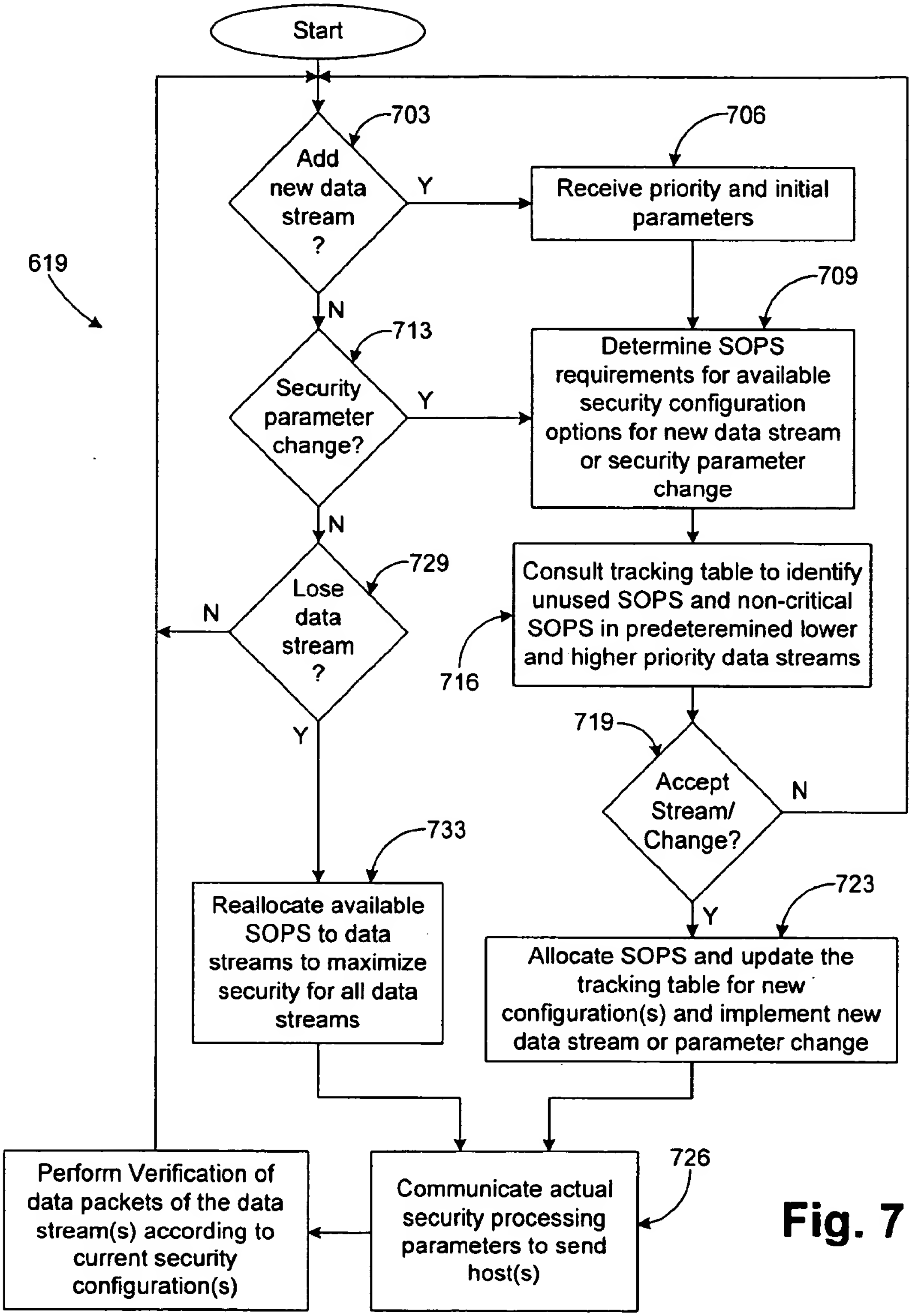

Start
703
Add new data stream ?
Y
N
706
Receive priority and initial parameters
709
Determine SOPS requirements for available security configuration options for new data stream or security parameter change
619
713
Security parameter change?
Y
N
729
Lose data stream ?
N
Y
Consult tracking table to identify unused SOPS and non-critical SOPS in predeteremined lower and higher priority data streams
716
719
Accept Stream/ Change?
N
Y
733
Reallocate available SOPS to data streams to maximize security for all data streams
723
Allocate SOPS and update the tracking table for new configuration(s) and implement new data stream or parameter change
726
Perform Verification of data packets of the data stream(s) according to current security configuration(s)
Communicate actual security processing parameters to send host(s)

Fig. 7

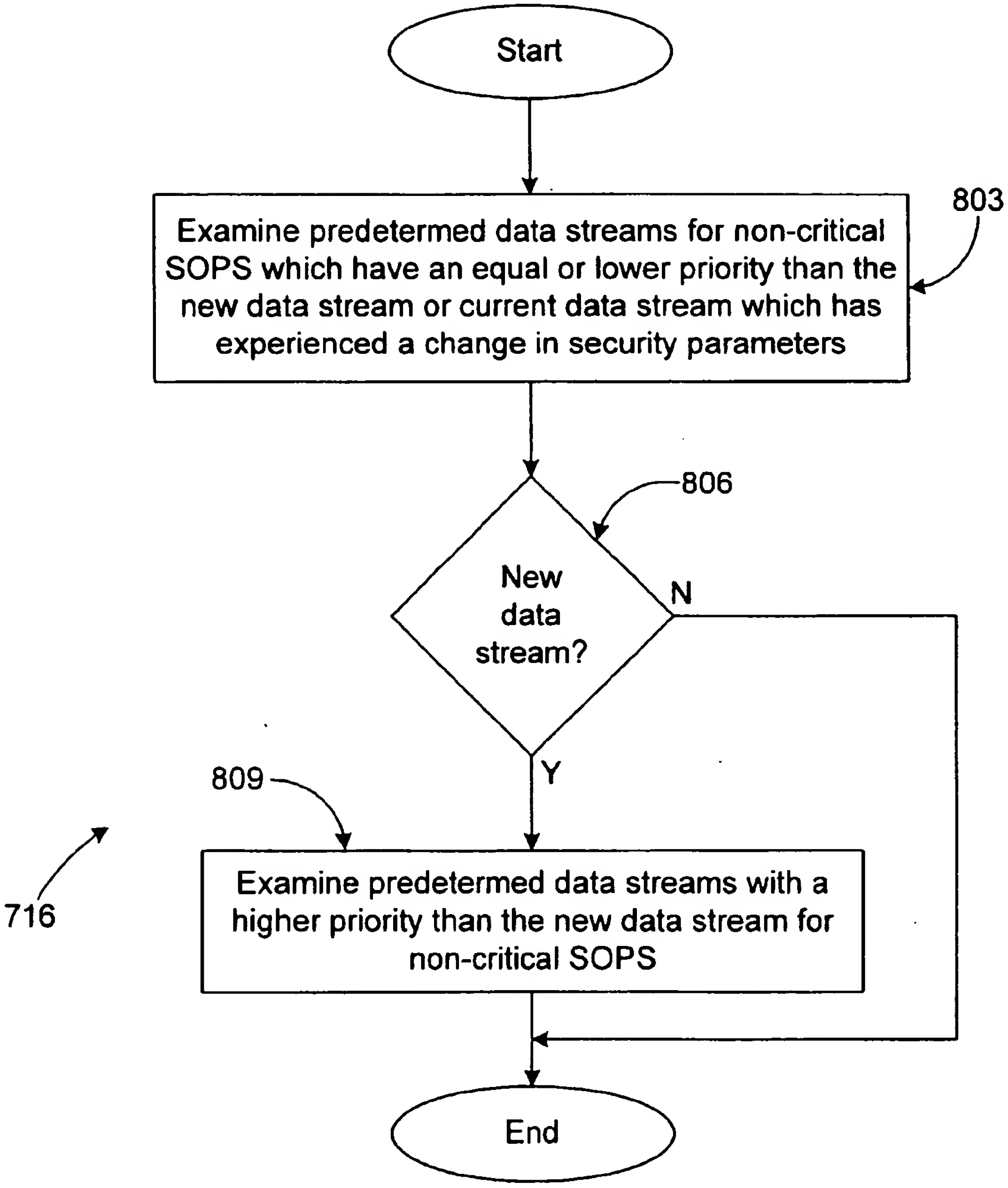

Start
Examine predetermed data streams for non-critical SOPS which have an equal or lower priority than the new data stream or current data stream which has experienced a change in security parameters
803
806
New data stream?
N
Y
809
Examine predetermed data streams with a higher priority than the new data stream for non-critical SOPS
716
End

Fig. 8

ADAPTIVE DATA SECURITY SYSTEM AND METHOD

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to and the benefit of U.S. Provisional patent application entitled "Authenticast: Adaptive Protocol to Enable Strong Authentication in High-Performance Applications" filed on Oct. 28, 1997 and accorded Ser. No. 60/063,551, which is incorporated herein by reference. This application is also a continuation Ser. No. 09/181,304 U.S. Pat. No. 6,108,583, having a priority date of Oct. 27, 1997, a filing date of Oct. 28, 1998, and issued on Aug. 22, 2000 to Schneck, et al. This application claims priority to and benefit of U.S. Pat. No. 6,108,583.

This application is a continuation of application Ser. No. 09,181,304, filed Oct. 28, 1997, now U.S. Pat. No. 6,108, 583.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

The U.S. government has a paid-up license in this invention and the right in limited circumstances to require the patent owner to license others on reasonable terms as provided for by the terms of DAAH04-96-1-0209 awarded by the U.S. Department of Defense.

TECHNICAL FIELD

The present invention is generally related to the field of data communications and, more particularly, is related to a system and method for securing data communication.

BACKGROUND OF THE INVENTION

Currently there is an exponential increase in the number of banks and other businesses that use the Internet to conduct transactions. The Internet is often a less expensive and less time consuming business medium than paper or the telephone. Electronic commerce and data interchange are increasing efficiency and giving companies a competitive edge in the global economy. With this growth in Internet electronic commerce, it becomes essential that greater security be provided for network-enabled transactions and collaboration.

The demand for information security is further elevated by the increasing prevalence of virtual private networks (VPNs), which are configurations by which private business is conducted over public media, such as the Internet. Sharing an existing public communications infrastructure is far more cost-effective than building a separate network for every business. However, security is required to create this "private" logical network over existing public wire. To create this VPN, security operations are invoked at both the source and destination nodes to ensure properties such as confidentiality, integrity, and authentication, for proof of origination and non-repudiation, of data.

Although some Internet commerce applications have been developed, they do not provide sufficiently strong security for trusted transfer of private data over a public medium. The very essence of strong security is the notion that the security medium employed to protect data cannot be compromised in a sufficiently short time to allow use or alteration of those data by an unauthorized party. Therefore, data protection mechanisms for strong security are required to be complex, and they thus have a high computation overhead which detracts from overall application performance. In the interest of performance, security procedures are often omitted. If Internet commerce applications are to succeed, they cannot compromise performance or security. In the best possible case, security mechanisms would be transparent to users. However, so far, security in the world wide web security is poor. It is relatively few vendors that can delivery invisible security. The inherent tradeoffs in realizing both security and performance comprise the challenge we face in providing them.

In addition, law enforcement officials are becoming increasingly dependent on the availability of real-time, network collaborative and shared applications. For example, police officers are assisted by real-time photos and data delivered directly to their vehicles. This often requires strong authentication measures which are admissible in court as proof of origin, identity, and integrity of certain data and electronic evidence. The ability to dynamically vary levels of authentication to match available resources and current requirements provides users of law enforcement applications options to employ strong security and use data as evidence while still receiving these data in a timely manner. This option was previously unavailable.

In addition, the healthcare industry is another example of a business relying heavily on shared or collaborative applications to provide greater customer service. For example, electronic communications infrastructures such as the Internet facilitate and expedite potentially worldwide collaboration on x-ray images or case studies. These materials, however, contain personal data, and for patient privacy and safety, are often required to be encrypted, for confidentiality and/or authenticated, for identification of the image. Again, security is necessary for these applications that enable the networked collaboration, yet the security could be detrimental if it hampers the speed with which the information can be used to help the patient.

SUMMARY OF THE INVENTION

The present invention provides a system and method for facilitating adaptive security between a send host and a receive host. Briefly described, in architecture, the system includes a send host having a processor coupled to a data bus, a memory coupled to the data bus, an input device coupled to the data bus to input a desired security configuration for a data stream to be communicated to a receive host, and an output device coupled to the data bus to display the desired and actual security configurations for the data stream on an output display, the actual security configuration generally being received from the receive host. The processor operates according to adaptive security logic stored on the memory which includes logic to generate a plurality of data packets associated with the data stream, the data packets including an authentication data block with an authentication header containing the actual security configuration and a signature.

The system further includes a receive host which comprises a processor, memory, data input, and data output, all coupled to a data bus. The data input is configured to receive at least one data stream comprising a number of data packets, the data packets including an authentication data block having an authentication header and a signature. The processor runs according to adaptive security logic stored on the memory. The adaptive security logic includes logic to decompose an authentication header in the data packets, logic to perform a variable percentage verification on the data packets from the data stream, and logic to determine an actual verification percentage performed based on a number of available security operations, a minimum verification threshold, and a desired verification target, the minimum verification threshold and the desired verification target being contained in the authentication header. The adaptive security logic also includes logic to verify the data packets using delayed verification techniques.

The present invention can also be viewed as providing a method for communicating a data stream employing adaptive security. In this regard, the method can be broadly summarized by the following steps:

- identifying a desired verification type, a desired security algorithm, a minimum security level, and a target security level in a send host for communicating a data stream from the send host to a receive host;
- determining an actual verification type, an actual security algorithm, and an actual security level in the receive host based on the desired verification type, desired security algorithm, minimum security level, target security level, and an availability of a number of security operations per second (SOPS);
- communicating the actual verification type, the actual security algorithm, and the actual security level from the receive host to the send host;
- generating a plurality of data packets associated with the data stream in the send host, the data packets having an authentication data block with an authentication header, the authentication header containing the actual verification type, actual security algorithm, minimum security level, target security level, and the actual security level;
- verifying the data packets on a percentage basis if the actual verification type is percentage based verification, the percentage based verification being performed at the actual security level which is greater or equal to the minimum security level and less than or equal to the target security level; and
- performing a delayed verification on the data packets if the actual verification type is delayed verification.

The present invention has numerous advantages, a few of which are delineated hereafter as merely examples. An advantage of the invention is that it facilitates more effective data security by allowing a receive host to adapt the security level at which a data stream is verified based upon the availability of host processor resources to provide security operations per second (SOPS) in the receive host. In this manner, data streams received by the receive host are not delayed or lost clue to a security processing bottleneck which can occur if multiple incoming data streams stress the security operation capacity of a particular receive host.

Other advantages of the invention include that it is user friendly, robust and reliable in operation, efficient in operation, and easily implemented for mass commercial production.

Other features and advantages of the present invention will become apparent to one with skill in the art upon examination of the following drawings and detailed description. It is intended that all such additional features and advantages be included herein within the scope of the present invention.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

The invention can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention.

Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

FIG. 1 is a functional block diagram of an data security system according to an embodiment of the present invention;

FIG. 2 is a block diagram of the send host of FIG. 1;

FIG. 3 is a flow chart showing the send host authentication logic of FIG. 2;

FIG. 4 is a drawing of the output display of FIG. 2;

FIG. 5 is a drawing of an authentication data block generated by the send host of FIG. 2;

FIG. 6 is block diagram of the receive host of FIG. 1;

FIG. 7 is a flow chart showing the receive host authentication logic of FIG. 6; and

FIG. 8 is a flow chart of a SOPS identification subroutine of FIG. 7.

DETAILED DESCRIPTION OF THE INVENTION

Turning to FIG. 1, shown is a functional block diagram of an authentication system **100** according to an embodiment of the present invention. The authentication system **100** includes at least one send host **103** and a receive host **106**. Although only a single send host **103** is shown, it is understood that multiple send hosts **103** may communicate with the receive host **106**, the single send host **103** being shown for purposes of the following discussion. Likewise, the single send host **103** may communicate with multiple receive hosts **106**, the single receive host **106** being shown for purposes of the following discussion as well. Additionally, multiple send hosts **103** may communicate with multiple receive hosts **106**.

The send host **103** generates a data block **109** or receives the data block **109** from a separate source to be communicated to the receive host **106**. Generally, the data block **109** is of a predetermined length and may be formed, for example, from a continuous data stream or data file received by the send host **103**. The data block **109** is formed as the data payload in a packet to be communicated to the receive host **106** as will be discussed.

The data block **109** may include any type of data such as, for example, but not limited to, audio, video, or other computer data. The send host **103** also includes a key **113** which may be a block of data of predetermined length as is known in the art. The key **113** may be a private key for signing the data, or a public key for encrypting the data as known in the art. Note that other keys may be employed to accommodate different authentication or encryption algorithms.

The send host **103** includes a signature generator **116** which generates a signature block **119** from the data block **109** and the key **113** using a predetermined security algorithm which may be, for example, but not limited to, a security algorithm such as the Digital Signature Algorithm (DSA), the Rivest-Shamir-Adleman (RSA) algorithm, or secret key authentication, which are generally known in the art.

The send host **103** also includes an authentication header generator **123** which generates an authentication header **126**. The authentication header **126** includes various data fields, such as, for example, an authentication sequence number, data frame size, frame type, security algorithm, verification type, minimum security level, target security level, and an actual security level. The receive host **106** employs these data fields to generate an actual security configuration to achieve authentication of a data stream communicated from the send host 103. The actual security configuration is dynamic in that it may be changed by either the send host 103 or the receive host 106 during the course of data communication therebetween in response to user or application requirements, or changes in computer resource availability as will be discussed.

The authentication header generator 123 may receive a desired security configuration from the user input 129 or a default desired configuration may be received from a default storage 133 for a particular data stream. The desired security configuration is displayed on a display device 136 along with the actual security configuration which may ultimately be determined by the receive host 106 depending upon the specific desired security configuration specified by the user. Upon system startup, the default desired security configuration is obtained from the default storage 133 and displayed on the display device 133. A user may then alter the desired security configuration via the user input 129.

The authentication header generator 123 generates the authentication header 126 which contains the actual security configuration to be placed in an authentication data block 139. Together, the data block 109 and the authentication data block 139 make up a data packet 143 which is communicated to the receive host 106. The data stream is thus a continuous stream of "signed" data packets 143, each data packet 143 containing an authentication data block 139 with an authentication header 126 and a signature block 119. It may also be possible, however, that the data stream may only contain a predetermined percentage of "signed" data packets 143 as desired by the user. The security configuration identified in the authentication header 126 is initially determined from the desired security specification and may be altered based on feedback received from the receive host 106. Note that the user may alter the desired security specification after data communication is commenced between the send host 103 and the receive host 106. The receive host 106 may also alter the actual security configuration based on the operating state of the receive host 106, the altered security configuration being displayed to the user on the display device 133.

The receive host 106 receives the data packet 143 and the authentication header 126 is decomposed in an authentication header decomposer 146 in which the above stated fields are separated from the data packet 143 for use by the receive host 106. The receive host 106 then attempts to execute the desired security verification configuration contained in the authentication header 126. The receive host 106 may employ one of several verification types. In the preferred embodiment, two specific verification types are used, namely, delayed authentication verification and percentage based verification, although other verification types may be employed as well.

When delayed authentication is employed, a predetermined number of signature blocks 119 and corresponding data blocks 109 from the data packets 143 received are collected in a bundle 149 as indicated by a first functional switch 153 which is placed in the D position. Thereafter, the delay authentication verifier 156 operates on the bundle 153 and verifies the data blocks 109 contained therein together using appropriate hashing functions known by those skilled in the art. Delayed verification will verify one hundred percent of the data packets. Once verified, the data blocks 119 are then provided to the receive data processor 159 for further processing according to the specific application. Delayed authentication is almost always available as a verification option, even when security processing resources are limited as delayed authentication exploits hashing techniques that reduce a large amount of data to a relatively small amount which can be verified rather quickly. However, there is a greater probability of processing delay due to data corruption as many data blocks are verified at once, which means that a single corrupted data block would require the entire data block to be retransmitted for verification.

When percentage based verification is employed, a predetermined percentage of signature blocks 119 and corresponding data blocks 109 from the data packets 143 are accessed by a percentage authentication verifier 163 as indicated by the first functional switch 149 being placed in the P position. A second functional switch 166 provides access to a particular signature block 119 and corresponding data block 109 upon which verification is performed when in the V position. Otherwise, when the second functional switch 166 is in the N position, the data block 109 is passed on to the receive data processor 159 without verification, the signature block 119 being discarded as shown. After a particular data block is verified by the percentage authentication verifier 163, the verified data block 109 is provided to the receive data processor for further processing according to the specific application. Note that the frequency or actual security level at which the second functional switch 166 provides access to the signature blocks 119 and corresponding data blocks 109 is determined by the security monitor 169. Generally, the security levels as discussed herein refer to the percentage of verified data packets in the receive host 106. The security monitor 169 also determines the verification type as indicated by the first functional switch 153, as well as the specific security algorithm employed by both the delayed authentication verifier 156 and the percentage authentication verifier 163.

The security monitor 169 attempts to specify an actual verification type, actual security algorithm, and an actual security level according to the desired security configuration received from the send host 103. However, the receive host 106 may not have enough processor time or security operations per second (SOPS) to provide the desired security configuration due to the verification of other data streams which currently employ much if not all of the SOPS available in the receive host 106 at a given moment. Consequently, the security monitor 169 may force a change in the verification type, security algorithm, and/or the actual security level that differs from the desired security configuration received by the send host 103 in order to accommodate the data stream.

In order to change the security algorithm employed, the security monitor 169 sends the new security algorithm to be employed to the send host 103 via a return path, the authentication header generator 123 implementing the new security algorithm while changing the authentication header to indicate the new security algorithm appropriately. The security algorithm is changed in this manner because the generation of the signature block 119 is performed by the send host 103.

Likewise, a change in the verification type is effected by the security monitor 169 by sending the new verification type to the send host 103 via the return path. The new verification type is then placed in the authentication header 126 by the authentication header generator 123. When a data packet 143 containing the new verification type reaches the receive host 103, then the security monitor causes the first functional switch 153 to move to the D position to employ delayed authentication verification in synch with the incoming data packets 143 earmarked for such verification type.

A change in the actual security level when percentage based verification is employed may occur in the receive host

106 or the send host 103. In the receive host 106, the actual security level is raised or lowered based upon the number of SOPS available in the receive host 106. It is understood that a lower security level requires a correspondingly lower number of SOPS to implement and vice versa. Note that the actual security level is not lowered below a predetermined minimum security level which is identified in the authentication header 126 so as to maintain a minimum amount of security. The actual security level determined by the security monitor 169 is communicated to the send host 103 for display on the display device 136.

The actual security level may be changed by the send host 103 by the user. Specifically, the user may adjust the actual security level via the user input 129. If the user adjusts the actual security level to a point which the receive host 106 is unable to maintain due to a lack of SOPS availability, the receive host 106 may generally react by switching to delayed verification in which one hundred percent of the data packets are verified as delayed verification can generally be performed with a minimal number of SOPS due to the hashing functions employed.

Note that when the receive host 106 alters any security parameter due to a lack of available SOPS, the receive host 106 may store the previous desired parameters in memory so that the receive host may revert back to the previous desired parameters when SOPS become available. These parameters may include, but are not limited to, the desired verification type and the desired security algorithm.

The receive host 106 includes a receive host input 173 in which a user may alter the actual security parameters manually. The receive host 106 displays the desired and actual security configuration on the receive display device 173 to be viewed by the user.

Note that the functionality of the send host 103 and the receive host 106 as described above and in the following discussion may be resident in a single computer system which may act as a send host 103 or a receive host 106 at any given time, depending upon whether the user is sending or receiving data. Further, a single computer system may simultaneously act as a send host 103 and a receive host 106 at the same time, communicating one or more data streams to a number of destination data endpoints and receiving one or more data streams from other origination data endpoints.

All of the above functionality discussed herein is implemented at a user/application level as known in the art which provides a distinct advantage as the present invention may be employed regardless of the underlying physical layer such as a network.

Referring next, to FIG. 2, shown is block diagram of the send host 103 according to an example embodiment of the present invention. The send host 103 includes a computer system 203 with a processor 206 and a memory 209 which are electrically coupled to a data bus 213. Also electrically coupled to the data bus 213 are a data storage device 216, an input interface 219, an output display interface 223, and a data communication interface 226. The input interface module 219 in turn is electrically coupled to a user input interface device 129 such as a keyboard, mouse, or other suitable device. Likewise, the output display interface 223 is electrically coupled to an output display device 136 such as a cathode ray tube (CRT) or suitable display device. The data storage device 216 may be a hard drive, floppy disk drive, fixed memory device, or other suitable means of data storage. The data communication interface 226 allows the send host 103 to communicate with the receive host 106 (FIG. 1) via a data communications channel (not shown). In performing the various tasks as discussed herein, the processor 206 operates according to the send host authentication logic 229 stored on the memory 209.

Turning next to FIG. 3, shown is flow chart which depicts the send host authentication logic 229. The send host authentication logic 229 begins with block 303 in which the desired security configuration is determined and displayed on the output display device 136 (FIG. 1). The desired security configuration may include the desired security algorithm, the desired verification type, the minimum security level, the target security level, and the actual security level. The actual security level may initially be set equal to the target security level until the receive host 103 alters the actual security level due to the lack of available processing resources to accomplish the target security level. These parameters may initially be read from a default security parameters file saved on the data storage device 216 (FIG. 2) or simply entered by the user via the user input interface device 129. Also, during startup, a data stream priority level is communicated from the send host 103 to the receive host 106 which is used by the receive host 106 in allocating processor resources or SOPS to the number of data streams received at any given moment. Note that the priority level may also be included as a data field in the authentication header 126 (FIG. 1) and may be altered by the user at the send host 103. The priority level is also displayed on the display device 136.

Next, in block 306, the send host 103 (FIG. 1) establishes a data communications link with the receive host 106 (FIG. 1) undergoing an initial training procedure in which the desired security parameters are communicated from the send host 103 to the receive host 106. The receive host 106 evaluates its capacity to verify the data packets 143 (FIG. 1) to be communicated according to the desired security configuration, and, if the receive host 106 has the necessary available SOPS, the verification of the data stream is performed according to the desired security configuration. If the requisite SOPS are not available, then the receive host 106 will determine and send an actual security configuration back to the send host 103 if the desired security configuration allows such parameters to be varied by the receive host 106. The actual security configuration may include, for example, the actual security algorithm, the actual verification type, and the actual security level. If the desired security configuration does not allow such changes, then the data link will be rejected by the receive host 106. The actual security parameters are then displayed on the output display device 130 (FIG. 2), if the data stream is accepted by the receive host 106.

The send host authentication logic 229 then progresses to block 309 in which the data packets 143 (FIG. 1) are assembled with the authentication data block 139 (FIG. 1) which includes the authentication header 126 (FIG. 1) and the signature block 119 (FIG. 1). The signature block 119 (FIG. 1) is generated using the actual security algorithm which is the same as the desired security algorithm specified in the desired security configuration unless altered by the receive host 106. The data packets 143 are communicated to the receive host 106.

Next in block 313, the send host authentication logic 229 determines whether the desired security configuration has been changed by the user via the user input interface device 129 (FIG. 2). If such a change has been made, then the send host authentication logic 229 progresses to block 316. If not, then the send host authentication logic 229 progresses to block 319. In block 319, the send host authentication logic 229 determines whether any of the actual security parameters have been changed by the receive host 319. If such a change has been made, then the send host authentication logic 229 moves to block 316. In block 316, the desired and actual security parameters displayed by the output display device 136 are altered to reflect any changes made. Thereafter, the send host authentication logic 229 reverts back to block 309 in which the data packets 143 are generated using the new security parameters. According to the preferred embodiment, the actual security level may be altered by the send host 103 as initiated by the user, for example, whereas the verification type and the security algorithm may not be changed by the send host 103 after the startup of data communication because the receive host 106 controls these parameters.

If in block 319, the receive host 106 does not change any of the actual security parameters, then the send host authentication logic 229 progresses to block 323 where it is determined whether the transmission of the data stream is complete. If the transmission is not complete, the send host authentication logic 229 reverts back to block 309 to continue to generate and communicate data packets. If the transmission of the data stream is complete in block 323, then the send host authentication logic 229 ends. Thus, the send host authentication logic 229 ultimately establishes an actual security configuration by which the data stream is communicated and reacts to any changes in the security parameters of the actual security configuration initiated by either the user or by the receive host 106. In this manner, the security configuration adapts over time to facilitate optimum data transmission speed while providing adequate security.

With reference to FIG. 4, shown is an output display screen 403 appearing on the output display device 136, which may be a CRT, for example, or other suitable display device or devices. The output display screen 403 includes a desired verification type block 406 in which one may toggle between delayed verification 409, percentage based verification 413, and automatic verification 416. Where delayed verification 409 or percentage based verification 413 are chosen, the receive host (FIG. 1) is forced to employ the desired verification type chosen and may not switch to an alternative verification type. Where automatic verification 416 is chosen, the actual verification type can be determined by the receive host (FIG. 1) based on availability of SOPS, etc. Preferably, the receive host 106 will attempt to establish percentage based verification before delayed verification due to a greater reliability and a lesser susceptibility to delays, when the desired security configuration allows the receiver to select the verification type. Generally, delayed verification is employed when percentage based verification cannot be accommodated by the receive host 106.

The output display screen 403 also includes a desired parameters block 419 which displays a desired security level range which includes a minimum security level 423 and a target security level 426 which may be entered with the user input interface 129 (FIG. 2) such as a keyboard for example. The desired parameters block 419 also includes a desired security algorithm 429 and a fixed block 433. The desired parameters block 419 may offer a pull down list of security algorithms within which one may chose a particular algorithm to be employed. The fixed block 433 indicates whether the receive host 106 may specify an actual security algorithm other than that chosen by the user as indicated by the desired security algorithm 429.

The output display screen 403 also includes a security thermostat 436 which includes a slide control 443 that indicates the actual security level 439 between the minimum and target security levels 423 and 426. Note that the slide control may be moved up and down with, for example, a mouse which guides a pointer on the output display screen. Next to the security thermostat 436 is an actual parameters block 446 which shows an actual security algorithm 453 and an actual verification type 456. The actual security algorithm 453 and the actual verification type 456 are those dictated by the receive host 106 (FIG. 1) based on SOPS availability. If enough SOPS are available to implement the desired parameters, then the parameters in the actual parameters block 446 would mirror the desired parameters in the desired verification type block 406 and the desired parameters block 419.

In addition, the output display screen 403 features a data stream identifier block 459 in which includes a current data stream indicator 463 with toggle buttons 466. The toggle buttons 466 increase or decrease the value in the current data stream indicator 463. The current data stream indicator 463 indicates the particular data stream for which parameters are displayed on the output display screen 403 at a given time in which the send host 103 is communicating two or more data streams to two or more receive hosts 106.

The output display screen 403 includes a default configuration save button 469 which causes the current desired security parameters as reflected in the desired verification type block 406, the desired parameters block 419, and the security thermostat 436 to be saved to the data storage device 216. In the preferred embodiment, this default configuration is employed whenever a new data stream is initiated, where the various default parameters may be altered as the user sees fit.

The output display screen further includes a packet signed percentage block 473 which indicates a percentage of data packets 109 (FIG. 1) for which a signature block 119 (FIG. 1) is generated. This value may be less than one hundred percent when processor resources are stressed in the send host 106 (FIG. 1), thereby reducing the demand for processor resources for the signature generation.

Finally, the output display screen features a priority selection block 476 with a priority indicator 479 and priority indicator toggle buttons 483. The priority of a particular data stream may be chosen by the user by manipulating the toggle buttons 483 with a button on a mouse (not shown). In this manner, one may alter the priority of the particular data stream.

Turning then, to FIG. 5, shown is the authentication data block 139. The authentication data block 139 includes the authentication header 126 with various data fields to communicate the various security parameters discussed previously as well as additional parameters. It is understood that the particular order and size of the data fields as shown herein is as an example as other sizes and orders may be employed. The authentication header 126 includes an authentication sequence number field 503 which uses bytes 0–16. The authentication sequence number field 503 is employed to keep track of the order in which data packets are authenticated and received. Next, a data frame size field 506 occupying bytes 17–20 is specified which indicates the size of the authentication data block 139. A frame type field 509 which occupies the 21st byte is specific to an encoding employed, for example, I, B, or P frames as in MPEG encoding, which is known in the art.

Next, a security algorithm field 513 is specified in byte 23 which indicates the actual security algorithm 513 employed by the receive host 106 (FIG. 1). In byte 24 is a verification type field 516 which indicates the actual verification type employed by the receive host 106. In byte 25, a security level minimum field 519 is defined which indicates the minimum security level or verification percentage to be performed by the receive host **106**. Note that the minimum security level can not be changed by the receive host **106** so that a minimum level of verification is maintained as desired by the user. Next is a target security level field **523** which occupies byte **25** and specifies the target security level. The target security level is set by the send host **103** while the receive host **106** attempts to meet this level. The target security level field **523** is followed by an actual security level field **526** which occupies byte **26** of the authentication data block **139**. The actual security level **526** may be determined by the receive host **106** in light of available processor resources, or the user at the send host **103** may manually change the actual security level **526** via the security thermostat **436**. Byte **27** is occupied by a priority field **529** which holds the actual priority assigned to the data stream. Finally, the signature block **119** follows the priority field **529** and is of variable length depending upon the particular security algorithm employed.

Turning next to FIG. **6**, shown is a block diagram of the receive host **106** (FIG. **1**). The receive host **106** is comprised of a computer system **603** which includes a processor **606**, a memory **609**, and a data communication interface **613**. The processor **606**, memory **609**, and data communication interface **613** are all electrically coupled to a data bus **616**. The processor **606** operates according to receive host verification logic **619** stored on the memory **609**. The data communication interface **613** is adapted to be electrically coupled to a number of channels **623** through which the receive host **106** may communicate with any number of send hosts **103**. The receive host **106** further includes a data storage device **626**, an input interface **629**, and an output display interface **633**, all of which are electrically coupled to the data bus **616**. The input interface **629** is also electrically coupled to receive host input interface device **173** such as a keyboard or mouse. Similarly, the output display interface **633** is electrically coupled to the receive display device **176** which may be a CRT or other similar device. The receive display device **176** features the output display screen **403** (FIG. **4**) to inform the end user of the operation of the receive host **106**.

Referring then, to FIG. **7**, shown is a flow chart which depicts the receive host verification logic **619**. The receive host verification logic **619** begins with block **703** in which it is ascertained whether a particular send host **103** (FIG. **1**) is attempting to establish secure data communication with the receive host **106** (FIG. **1**). If so, the receive host verification logic **619** progresses to block **706** in which in which the receive host **106** is provided with the priority value for the data stream and the desired parameters including the security algorithm, verification type, minimum and target security levels, and an initial actual security level which may equal, for example, the target security level. Thereafter, the receive host verification logic **619** proceeds to block **709**.

If in block **703** there is no new data stream to be received, then the receive host verification logic **619** proceeds to block **713** in which it is determined if any of the desired security parameters, specifically the actual security level, has been changed by the user at the send host **103**. If any security parameters have changed, then the receive host verification logic **619** moves to block **709**.

In block **709**, the receive host verification logic **619** evaluates either the potential new data stream based on the parameters received in block **706**, or the change in the actual security level or other security parameters detected in block **713** to determine how many SOPS are required by the new data stream or the security parameter change in an existing data stream. Generally, such information is stored in a tracking table in the memory **609** that may include values which indicate the data stream priority, an amount of SOPS necessary to maintain the minimum security level, the amount of SOPS consumed to maintain the actual security level, and the amount of SOPS necessary to achieve the target security level for each existing data stream received by the receive host. The tracking table may also be stored on the data storage device **626** or other suitable storage device.

Thereafter, the receive host verification logic **619** progresses to block **716** where the tracking table is consulted to determine how many SOPS are available to accommodate the potential new data stream or the desired change in the actual security level. In particular, the receive host verification logic **619** identifies how many unused SOPS are available and how many non-critical SOPS may be diverted from the verification processing of other data streams to facilitate the potential new data stream or the change in the actual security level. Non-critical SOPS are those used to perform a percentage based verification at an actual security level which is greater than the minimum security level for a particular data stream. That is to say, non-critical SOPS may be diverted from the verification processing of a particular data stream and the minimum security level can be maintained for that data stream.

The receive host verification logic **619** then progresses to block **719** in which it is determined whether there are enough unused SOPS and non-critical SOPS as indicated by the tracking table which may be diverted to accommodate the new data stream or the security parameter change. If such is the case, then the receive host verification logic **619** proceeds to block **723**. If not, then the new data stream is rejected and/or the security parameter change is not implemented and the receive host verification logic **619** reverts back to block **703**. For example, if one attempts to increase the actual security level by manipulating the security thermostat **436** (FIG. **4**), then the receive host **106** will attempt to facilitate the increase in the actual security level. If the receive host **106** cannot achieve the higher security level using percentage based verification, then the receive host **106** may automatically switch the verification type to delayed verification to accommodate a security level of one hundred percent.

In block **723**, the previously identified non-critical SOPS and any unused SOPS are diverted to accommodate the new data stream and/or the security parameter change. The tracking table is updated with the new allocation for each altered data stream including the new data stream if one is implemented. Thereafter, the receive host verification logic **723** progresses to block **726**.

Referring back at block **713**, if there is not change to the security parameters, then the receive host verification logic **619** proceeds to block **729** in which it is determined whether the communication of any current data stream has terminated. If such is not the case, then the receive host verification logic **619** reverts back to block **703**. If a current data stream has ceased communication in block **729**, then the receive host verification logic **619** progresses to block **733**. In block **733**, the SOPS which were employed in processing the now terminated data stream are reallocated to the existing data streams to maximize security for all of the data streams. Thereafter, the receive host verification logic **619** continues to block **726**.

In block **726**, the security parameters for all data streams which are new or altered due to the allocation or reallocation of the SOPS in blocks **723** and **733** are communicated to their respective send host(s) 103. Next, in block 736, the verification of the data packets of the current data streams are performed according to the security parameters determined for each data stream. Thereafter, the receive host verification logic 619 reverts back to block 703.

Note that the receive host verification logic 619 operates in a continuous loop searching for changes in the status quo of the processing of the data streams and reacts to changes by either reallocating processor resources (SOPS) to accommodate a change, or rejecting such changes altogether and maintaining the status quo.

Finally, referring to FIG. 8, shown is a flow chart of the non-critical SOPS identification subroutine 716. The subroutine 716 executes the logical steps taken in identifying non-critical SOPS with which to accommodate a change in security parameters of an existing data stream or to accommodate a new data stream. Beginning with block 803, the resource tracking table is consulted looking for predetermined existing data streams with a priority that is equal to or lower than the priority of the new data stream or the changed data stream are examined to determine the quantity of non-critical SOPS in each. The predetermined number of lower priority data streams may be, for example, a predefined number of data streams starting from the lowest priority up, or the predetermined number of lower priority data streams may be determined at random. The predetermined number of data streams examined may include all of the lower priority data streams if there are not too many to examine.

Next, in block 806 if a new data stream is sought to be implemented, then the subroutine 716 progresses to block 809. If a new data stream is not to be implemented in block 806, then the subroutine 716 ends. In block 809, predetermined data streams with a higher priority than the new data stream are examined for non-critical SOPS. The predetermined data streams examined may be, for example, a specific number of data streams starting from the highest priority down, or a random sampling of the higher priority data streams. The predetermined number of data streams examined may include all of the higher priority data streams if there are not too many to examine within an acceptable time period. Thereafter, the subroutine ends.

Note that both the send host authentication logic 229 and the receive host verification logic 619 of the present invention can be implemented in hardware, software, firmware, or a combination thereof. In the preferred embodiment(s), both the send host authentication logic 229 and the receive host verification logic 619 are implemented in software or firmware that is stored in a memory and that is executed by a suitable instruction execution system.

The flow charts of FIGS. 3, 7, and 8 show the architecture, functionality, and operation of a possible implementation of the adaptive security software employed by the send host 103 (FIG. 2) and the receive host 106 (FIG. 6). In this regard, each block represents a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that in some alternative implementations, the functions noted in the blocks may occur out of the order noted in FIGS. 3, 7, and 8. For example, two blocks shown in succession in FIGS. 3, 7, and 8 may in fact be executed substantially concurrently or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved, as will be further clarified hereinbelow.

In addition, the send host authentication logic 229 (FIG. 3) and the receive host verification logic 619 (FIGS. 7 and 8), each of which comprise an ordered listing of executable instructions for implementing logical functions, can be embodied in any computer-readable medium for use by or in connection with an instruction execution system, apparatus, or device, such as a computer-based system, processor-containing system, or other system that can fetch the instructions from the instruction execution system, apparatus, or device and execute the instructions. In the context of this document, a "computer-readable medium" can be any means that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device. The computer readable medium can be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a nonexhaustive list) of the computer-readable medium would include the following: an electrical connection (electronic) having one or more wires, a portable computer diskette (magnetic), a random access memory (RAM) (magnetic), a read-only memory (ROM) (magnetic), an erasable programmable read-only memory (EPROM or Flash memory) (magnetic), an optical fiber (optical), and a portable compact disc read-only memory (CDROM) (optical). Note that the computer-readable medium could even be paper or another suitable medium upon which the program is printed, as the program can be electronically captured, via for instance optical scanning of the paper or other medium, then compiled, interpreted or otherwise processed in a suitable manner if necessary, and then stored in a computer memory.

Many variations and modifications may be made to the above-described embodiment(s) of the invention without departing substantially from the spirit and principles of the invention. All such modifications and variations are intended to be included herein within the scope of the present invention.

Therefore, having thus described the invention, at least the following is claimed:

1. A send host employing adaptive security, comprising:

a processor coupled to a data bus;

a memory coupled to the data bus;

an input device coupled to the data bus to input a desired security configuration for a data stream to be communicated to a receive host;

an output device coupled to the data bus to display an actual security configuration for the data stream, the actual security configuration being received from the receive host; and

adaptive security logic stored on the memory, the adaptive security logic including logic to generate a plurality of data packets associated with the data stream, the data packets including an authentication data block with an authentication header containing the actual security configuration and a signature.

2. The send host of claim 1, wherein the adaptive security logic further comprises security thermostat logic to control an actual security level, the actual security level being included in the authentication header.

3. The send host of claim 1, wherein the adaptive security logic further comprises logic to place a minimum verification percentage in the authentication header.

4. The send host of claim 1, wherein the desired security configuration is displayed on the output device.

5. The send host of claim 1, wherein the actual security configuration is displayed on the output device.

6. The send host of claim 1, wherein the desired security configuration comprises a desired verification type, a mini-

mum verification percentage, a target verification percentage, a security algorithm, and an actual verification percentage.

7. A receive host employing adaptive security, comprising:

a processor coupled to a data bus;

a memory coupled to the data bus;

a data communications interface coupled to the data bus, the data communications interface being configured to receive at least one data stream comprising a number of data packets, the data packets including an authentication data block, the authentication data block having an authentication header and a signature;

adaptive security logic stored on the memory, the adaptive security logic including logic including

logic to decompose the authentication header in the data packets;

logic to perform a variable percentage verification on the data packets from the data stream; and

logic to determine an actual verification percentage performed based on a number of available security operations in the receive host, a minimum verification percentage, and a target verification percentage, the minimum verification percentage and the target verification percentage being contained in the authentication header.

8. The receive host of claim 7, wherein the logic to determine an actual verification percentage further comprises logic to determine the processor time availability by examining the at least one data stream received by the input device for a non-critical processor time usage.

9. The receive host of claim 7, wherein the adaptive security logic further comprises:

logic to perform a delayed verification on a bundle of data packets from the data stream; and

logic to enable one of the delayed verification and the variable percentage verification based on a verification type field contained in the authentication header and on a number of available security operations in the receive host.

10. The receive host of claim 7, wherein the adaptive security logic further comprises logic to maintain a resource tracking table which indicates the security operations required to accomplish the minimum security level, the target security level, the actual security level, and the priority level of a particular data stream.

11. A send host employing adaptive security, comprising:

means for inputting a desired security configuration for a data stream to be communicated to a receiver;

means for displaying the desired security configuration and an actual security configuration for the data stream, the actual security configuration being received from the receiver; and

means for generating a plurality of data packets associated with the data stream, the data packets including a data block and an authentication data block having an authentication header containing the actual security configuration and a signature.

12. A receive host employing adaptive security, comprising:

means for receiving at least one data stream comprising a number of data packets, the data packets including an authentication data block, the authentication data block having an authentication header and a signature;

means for decomposing the authentication header in the data packets;

means for performing a percentage based verification on the data packets from the data stream;

means for determining an actual security level performed based on a number of available security operations, a minimum security level, and a target security level, and a desired actual security level, the minimum security level and the target security level being contained in the authentication header; and

means for communicating the actual security level to a send host.

13. The receive host of claim 12, wherein the means for determining an actual security level further comprises means for determining a processor time availability by examining a resource tracking table for a non-critical processor time usage of at least one existing data stream.

14. The receive host of claim 12, further comprising:

means for performing a delayed verification on a bundle of data packets from the data stream; and

means for enabling one of the delayed verification and the variable percentage verification based on a verification type field contained in the authentication header and on a number of available security operations in the receive host.

15. A method for communicating a data stream employing adaptive security, comprising the steps of:

identifying a desired verification type, a desired security algorithm, a minimum security level, a target security level, and a desired actual security level in a send host for communicating a data stream from the send host to a receive host;

determining an actual verification type, an actual security algorithm, and an actual security level in the receive host based on the desired verification type, desired security algorithm, minimum security level, target security level, and an availability of a number of security processor operations;

communicating the actual verification type, the actual security algorithm, and the actual security level from the receive host to the send host;

generating a plurality of data packets associated with the data stream in the send host, the data packets having an authentication data block with an authentication header, the authentication header containing the actual verification type, actual security algorithm, minimum security level, the target security level, and the actual security level;

verifying the data packets using percentage based verification if the actual verification type is percentage based verification, the percentage based verification being performed at the actual security level which is greater or equal to the minimum security level and less than or equal to the target security level; and

performing a delayed verification on the data packets if the actual verification type is delayed verification.

16. The method of claim 15, further comprising the step of altering the actual security level in the send host using a security thermostat.

17. The method of claim 15, further comprising the step of identifying the availability of a number of security operations per second (SOPS) by identifying a number of non-critical SOPS employed by a plurality of second data streams received by the receive host, and by identifying a number of unused SOPS in the receive host.

18. A computer program embodied on a computer-readable medium for operation in a send host to facilitate data communication with adaptive security, comprising:

- logic to input a desired security configuration for a data stream to be communicated to a receiver;
- logic to display a desired security configuration and an actual security configuration for the data stream, the actual security configuration being received from the receiver; and
- logic to generate a plurality of data packets associated with the data stream, the data packets including an authentication data block having an authentication header containing the actual security configuration and a signature.

19. A computer program embodied on a computer-readable medium for operation in a receive host to facilitate data communication with adaptive security, comprising:

- logic to receive at least one data stream comprising a number of data packets, the data packets including an authentication data block, the authentication data block having an authentication header and a signature;
- logic to decompose the authentication header in the data packets;
- logic to perform a percentage based verification on the data packets from the data stream;
- logic to determine an actual security level performed based on a number of aVailable security operations in a receive host, a minimum security level, and a target security level, the minimum security level and the target security level being contained in the authentication header; and
- logic to communicate the actual security level to a send host.

20. The computer program embodied on a computer-readable medium of claim **19**, wherein the logic to determine the actual security level further comprises logic to determine a processor time availability by examining a resource tracking table for a non-critical processor time usage of at least one existing data stream.

21. The computer program embodied on a computer-readable medium of claim **19**, further comprising:

- logic to perform a delayed verification on a bundle of data packets from the data stream; and
- logic to enable one of the delayed verification and the variable percentage verification based on a verification type field contained in the authentication header and on the number of available security operations in the receive host.

22. A computer program embodied in a modulated data signal for transmission across a network, the computer program being for operation in a send host to facilitate data communication with adaptive security, comprising:

- logic to input a desired security configuration for a data stream to be communicated to a receive host;
- logic to display the desired security configuration and an actual security configuration for the data stream, the actual security configuration being received from the receive host; and
- logic to generate a plurality of data packets associated with the data stream, the data packets including an authentication data block having an authentication header containing the actual security configuration and a signature.

23. A computer program embodied in a modulated data signal for transmission across a network, the computer program being for operation in a receive host to facilitate data communication with adaptive security, comprising:

- logic to receive at least one data stream comprising a number of data packets, the data packets including an authentication data block, the authentication data block having an authentication header and a signature;
- logic to decompose the authentication header in the data packets;
- logic to perform a percentage based verification on the data packets from the data stream;
- logic to determine an actual security level performed based on a number of available security operations, a minimum security level, and a target security level, the minimum security level, the target security level being contained in the authentication header; and
- logic to communicate the actual security level to a send host.

24. The computer program embodied in a modulated data signal of claim **23**, wherein the logic to determine an actual security level further comprises logic to determine a processor time availability by examining said at least one data stream received by the input device for a non-critical processor time usage.

25. The computer program embodied in a modulated data signal of claim **23**, further comprising:

- logic to perform a delayed verification on a bundle of data packets from the data stream; and
- logic to enable one of the delayed verification and the variable percentage verification based on an availability of computer resources and on a verification type in the authentication header.

26. A receive host employing adaptive security with respect to at least one data stream having a number of data packets received by the receive host, comprising:

- means for determining a number of available security operations in the receive host; and
- means for allocating the number of available security operations in the receive host to perform a verification of a number of the data packets in the at least one data stream.

27. The receive host of claim **26**, wherein the means for determining a number of available security operations in the receive host further comprises means for determining the number of available security operations based upon a priority assigned to the at least one data stream.

28. The receive host of claim **26**, wherein the means for determining a number of available security operations in the receive host further comprises means for determining a number of non-critical security operations.

29. A method for employing adaptive security with respect to at least one data stream having a number of data packets received by a receive host, comprising the steps of:

- determining a number of available security operations in the receive host; and
- allocating the number of available security operations in the receive host to perform a verification of a number of the data packets in the at least one data stream.

30. The method of claim **29**, wherein the step of determining a number of available security operations in the receive host further comprises the step of determining the number of available security operations based upon a priority assigned to the at least one data stream.

31. The method of claim **29**, wherein the step of determining a number of available security operations in the receive host further comprises the step of determining a number of non-critical security operations.

* * * * *